UNIVERSIDAD DE ALMERÍA

Departamento de Arquitectura de Computadores y Electrónica

Meta-heurísticas Híbridas para Optimización

Mono-objetivo y Multi-objetivo.

Paralelización y Aplicaciones

TESIS DOCTORAL

Raúl Baños Navarro

Almería, Diciembre 2006

UNIVERSIDAD DE ALMERÍA

Departamento de Arquitectura de Computadores y Electrónica

Meta-heurísticas Híbridas para Optimización Mono-objetivo y Multi-objetivo. Paralelización y Aplicaciones

TESIS DOCTORAL

Realizada por: **Raúl Baños Navarro**

para optar al grado de Doctor en Informática

Dirigida por: **Dra. Consolación Gil Montoya**

Almería, Diciembre 2006

Dña. Consolación Gil Montoya, profesora titular del Departamento de Arquitectura de Computadores y Electrónica de la Universidad de Almería

CERTIFICA: La presente memoria titulada *Meta-heurísticas Híbridas para Optimización Mono-objetivo y Multi-objetivo. Paralelización y Aplicaciones*, ha sido realizada por D. Raúl Baños Navarro bajo mi dirección en el citado departamento en cumplimiento de los requisitos para optar al grado de Doctor en Informática por la Universidad de Almería.

Almería, 3 de Noviembre de 2006

Dra. **Consolación Gil Montoya** — Directora de la tesis y tutora

Raúl Baños Navarro — Autor de la tesis doctoral

UNIVERSIDAD DE ALMERÍA

Departamento de Arquitectura de Computadores y Electrónica

Meta-heurísticas Híbridas para Optimización

Mono-objetivo y Multi-objetivo.

Paralelización y Aplicaciones

TESIS DOCTORAL

Raúl Baños Navarro

Agradecimientos

En estas líneas quiero expresar mi agradecimiento a *Consolación Gil*, por el tiempo y esfuerzo dedicado durante los últimos años como directora de esta tesis doctoral. Sin duda alguna, sus consejos y motivación han sido determinantes para el buen fin de la misma.

A todos los miembros del Departamento de Arquitectura de Computadores y Electrónica de la Universidad de Almería, y en especial a aquellos pertenecientes al grupo de investigación Supercomputación-Algoritmos dirigido por *Inmaculada García Fernández*.

A todos los que han trabajado directamente en temas de investigación relacionados con esta tesis, en especial a *Julio Ortega*, *María Dolores Gil*, *Antonio López Márquez*, *Francisco Gil*, *Juan Reca*, *Juan Martínez*, *Alfredo Alcayde*, *Julio Gómez*, y *José Ignacio Agulleiro*.

A todos los que hemos compartido el rol de becarios o técnicos del departamento, *Siham*, *Sebas*, *Bogy*, *Fran*, *Juani*. En particular a *Manu* y *Román* por su amistad desinteresada.

A *Ester*, *Miguel Cobo*, *Emilio del Castillo*, y *Miguel Ángel Palma*, que como secretaria y jefes de negociado han gestionado eficientemente innumerables trámites burocráticos.

A los miembros de la School of Computing de la Universidad de Napier, en Edimburgo. En especial al profesor *Ben Paechter* por sus consejos como supervisor durante mis estancias realizadas en dicho centro. A *Manuel López-Ibañez* por sus interesantes comentarios y sugerencias sobre optimización multi-objetivo. A *Jennifer Willies, Despina Davoudani, Denitsa Petrova, y Bart Craenen* por su hospitalidad en Napier.

A los miembros del EPCC (Edinburgh Parallel Computing Centre), por sus gestiones y hospitalidad durante mis estancias en Edimburgo gracias al programa HPC-Europa. En especial a *Catherine Inglis* y a *Chris Johnson*.

A las entidades y organismos que han subvencionado esta tesis doctoral. Al grupo de investigación Supercomputación-Algoritmos de la Universidad de Almería, a la Consejería de Innovación Ciencia y Empresa, al Ministerio de Ciencia y Tecnología por sus proyectos TIC2002-00228, y TIN2005-00447, y a la Comisión Europea por la financiación de estancias en la School of Computing de la Universidad de Napier (Edimburgo) mediante el proyecto RII3-CT-2003-506079.

A Faustino y María del Carmen

La ilusión vale cuando la realidad la toma de la mano (anónimo)

Lista de publicaciones

Capítulos de libros internacionales/Inter. book chapters

• **R. Baños**, C. Gil, M.G. Montoya, J. Ortega: *Adapting Multi-Objective Meta-Heuristics for Graph Partitioning*, In Applied Soft Computing Technologies: The Challenge of Complexity, Series: Advances in Soft Computing, A. Abraham y col. (eds.), Springer, 2006, pp. 123-132. ISSN: 1434-9922, ISBN: 3-540-31649-3.

• **R. Baños**, C. Gil, J. Gómez, J. Ortega: *Performance Analysis of Parallel Strategies for Bi-objective Network Partitioning*, In Applications of Soft Computing: Recent Trends, Series: Advances in Soft Computing, A. Tiwari y col. (eds.) Springer, 2006, pp. 291-300. ISSN: 1434-9992, ISBN: 3-540-29123-7.

• **R. Baños**, C. Gil: *Graph-Mesh Partitioning: An Overview of the Current State-of-the-art*, In Mesh Partitioning Tecnhiques and Domain Decomposition Methods, F. Magoules (ed.), Saxe-Coburg Publications (en prensa).

Revistas internacionales/International journals

• C. Gil, J. Ortega, M.G. Montoya, **R. Baños**: *A Mixed Heuristic for Circuit Partitioning*. Computational Optimization and Applications Journal 23(3), 2002, pp. 321-340.

• **R. Baños**, C. Gil, J. Ortega, F.G. Montoya: *Multilevel Heuristic Algorithm for Graph Partitioning*, (Raidl et al. eds.) Springer-Verlag Lecture Notes in Computer Science 2611, 2003, pp. 143-153.

• **R. Baños**, C. Gil, J. Ortega, F.G. Montoya: *Optimising Graphs Partitions using Parallel Evolution*, (Liardet et al. eds.) Springer-Verlag Lecture Notes in Computer Science 2936, 2004, pp. 91-102.

• **R. Baños**, C. Gil, J. Ortega, F.G. Montoya: *A Parallel Multilevel Metaheuristic for Graph Partitioning*, Journal of Heuristics 10(3), 2004, pp. 315-336.

• **R. Baños**, C. Gil, M.G. Montoya, J. Ortega: *A New Pareto-based Algorithm for Muti-Objective Graph Partitioning*, (Aykanat et al. eds.) Springer-Verlag Lecture Notes in Computer Science 3280, 2004, pp. 779-788.

• C. Gil, **R. Baños**, M.G. Montoya, J. Gómez: *Performance of Simulated Annealing, Tabu Search, and Evolutionary Algorithms for Multi-objective Network Partitioning*, Algorithmic Operations Research 1(1), 2006, pp. 55-64.

• **R. Baños**, C. Gil, B. Paechter, J. Ortega: *Parallelization of Population-based Multi-Objective Metaheuristics: An Empirical Study*, Applied Mathematical Modelling 30(7), 2006, pp. 578-592.

• **R. Baños**, C. Gil, B. Paechter, J. Ortega: *A Hybrid Meta-heuristic for Multi-objective Optimization: MOSATS*, Journal of Mathematical Modelling and Algorithms, 2006, DOI: 10.1007/s10852-006-9041-6.

• **R. Baños** , C. Gil, J. Reca, J. Martínez: *Implementation of Scatter Search for Multi-Objective Optimization: A Comparative Study*, Computational Optimization and Applications J. (aceptado).

Congresos Internacionales/International conferences

• **R. Baños**, C. Gil, J. Ortega, M.G. Montoya: *A Parallel Evolutionary Algorithm for Circuit Partitioning*. 11th Euromicro Conference on Parallel, Distributed and Network-based Processing. Published by the IEEE Computer Society Press, pp. 365-371. Genova (Italia), Febrero 2003.

• **R. Baños**, C. Gil, J. Ortega, F.G. Montoya: *Parallel Heuristic Search in Multilevel Graph Partitioning*. 12th Euromicro Conference on Parallel, Distributed and Network-based Processing. Published by the IEEE Computer Society Press, pp. 88-95. La Coruña (España), Febrero 2004.

• **R. Baños**: *Multi-Objective Meta-Heuristic Algorithms for NP-complete Problems*. HPC-Europa - TAM Meeting: Science and Supercomputing in Europe. Report 2004, pp. 149-151. Stuttgart (Alemania), Septiembre 2005.

Congresos Nacionales/Spanish conferences

• **R. Baños**, C. Gil, J. Ortega, M.G. Montoya: *Algoritmo Evolutivo Paralelo para Partición de Circuitos*. XIII Jornadas de Paralelismo, pp.141-146. Lérida, 2002.

• **R. Baños**, C. Gil, J. Ortega, F.G. Montoya: *Partición en Grafos mediante Optimización Evolutiva Paralela*. XIV Jornadas de Paralelismo, pp. 245-250. Leganés, 2003.

• **R. Baños**, C. Gil, J. Ortega, F.G. Montoya: *PMSATS: Metaheurística Multinivel Paralela para Partición de Grafos*. III Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, pp. 380-388. Cordoba, 2004.

• **R. Baños**, C. Gil, M.G. Montoya, J. Gómez: *Balanceo de Carga y Minimización de Comunicación en Redes*. I Congreso Español de Informática - XVI Jornadas de Paralelismo, pp. 655-661. Granada, Septiembre 2005.

• **R. Baños**, C. Gil, J. Gómez, J. Ortega: *MOSATS: Metaheurística Híbrida para Optimización Multi-Objetivo*. I Congreso Español de Informática - IV Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, pp. 137-144. Granada, Septiembre 2005.

• **R. Baños**, C. Gil, A. Márquez, J. Gómez: *Paralelización de Meta-heurísticas Multi-objetivo Poblacionales Basadas en Búsqueda Local*, XIV Jornadas de Paralelismo, pp. 413-418. Albacete, Septiembre 2006.

Notación utilizada

α, a, b: parámetros de la ecuación de Hazen-Williams.

γ: peso específico del agua.

ϵ: coeficiente de rugosidad en la ecuación de Darcy-Weisbach.

$\lambda_2(L(G))$: segundo autovalor de la matriz laplaciana.

μ: factor utilizado para calcular el número de niveles de agrupamiento en el esquema multi-nivel.

ρ_{cruce}: probabilidad de cruce.

ρ_{mut}: probabilidad de mutación.

σ_k: peso de ponderación del k-ésimo objetivo.

φ: factor de distorsión en ILSSA.

ψ: longitud de la memoria de la lista tabú.

$\triangle$: incremento en el coste de las soluciones (el valor 1 significa incremento 0 %).

$\wedge$: medida de área de Lebesgue.

C: coeficiente de rugosidad en la ecuación de Hazen-Williams.

c_i: coste, por unidad de longitud, de las tuberías de diámetro i.

D: espacio de decisión.

D_i: dominio de la i-ésima variable.

DB_{max}: desequilibrio máximo admitido.

DW: ecuación de Darcy-Weisbach.

d_i: diámetro de la i-ésima tubería.

E: conjunto de aristas de un grafo.

$|E|$: número de aristas del grafo.

$e_{i,j}$: arista que une los vértices i y j.

$ext(v_i)$: suma de pesos de aristas que conectan a v_i con vértices de sub-grafos diferentes.

F: espacio de búsqueda, espacio de soluciones, o conjunto factible.

f: función objetivo.

f_f: factor de fricción de la ecuación de Darcy-Weisbach.

G: grafo.

$gain(v_i,v_j)$: decremento en el número de cortes obtenido tras intercambiar de sub-grafos los vértices v_i y v_j .

H: hiper-volumen.

HW: valor de la ecuación de Hazen-Williams.

H_{tq}: elevación del nivel de agua en el tq-ésimo tanque.

h_j: presión de cabecera actual computada en el nodo j.

hc_i: hiper-cubo i.

hr_j: presión de cabecera requerida en el nodo j.

I_c: coste económico de la red.

I_r: índice de fialibilidad de la red.

$int(v_i)$: suma de pesos de aristas que conectan a v_i con vértices del mismo sub-grafo.

K: número de objetivos.

K_m: coeficiente multiplicativo para calcular n_e en el problema de redes de distribución de agua.

K_p: constante de penalización del problema de distribución de agua.

k: k-ésimo objetivo.

kv: kv-ésimo vecindario.

$L(G)$: matriz laplaciana del grafo G.

L_i: longitud total de las tuberías de diámetro i en la red.

$L(i,j)$: valor de la fila i, columna j de la matriz laplaciana del grafo G.

LT: lista tabú.

M: suma de pesos del sub-grafo más grande.

mR: ratio de migración.

$N(s)$: vecindario de la solución s.

$N^*(s)$: vecindario reducido de la solución s.

NC: número de comunicaciones de cooperación.

ND: archivo externo de soluciones no dominadas.

$ND(S)$: soluciones no dominadas de S.

NE: número de ejecuciones.

n_b: número de bombas de la red de distribución de agua.

n_d: número de diametros de tuberías disponibles en la red de distribución de agua.

n_e: número de evaluaciones de la función objetivo (condición de parada).

n_i: número de iteraciones de la función objetivo (condición de parada).

n_l: número de enlaces / tuberías de la red de distribución de agua.

n_r: número de reservas de la red de distribución de agua.

n_n: número de nodos de la red de distribución de agua.

np: número de procesadores.

P,P',P": conjunto o población de soluciones (equivalente a S, S', S").

P_{tam}: tamaño de la población.

PR: conjunto de referencia en búsqueda dispersa.

PR_{tam}: tamaño del conjunto de referencia.

PR': conjunto de combinaciones en búsqueda dispersa.

PR'_{tam}: tamaño del conjunto de combinaciones en búsqueda dispersa.

PW_l: potencia introducida por la l-ésima bomba en el sistema.

Q_{tq}: flujo que sale del tq-ésimo tanque.

qq_i: caudal de la tubería i.

q_j: flujo del nodo j.

S,S',S": conjuntos de soluciones (equivalente a P, P', P").

SC: métrica de cobertura de conjuntos.

SG: número de sub-grafos a dividir un grafo.

$Sub(v_i)$: sub-grafo al que pertenece v_i.

sg: sg-ésimo sub-grafo.

s,s',s": solución, o individuo.

s_{ref}: solución imaginaria de referencia para calcular el hiper-volumen (H).

t: valor actual de la temperatura para el enfriamiento simulado.

T_{enfr}: factor de enfriamiento en la temperatura para enfriamiento simulado.

T_i: temperatura inicial para enfriamiento simulado.

T_{parada}: temperatura de parada para enfriamiento simulado.

u_i: dominio de la variable x_i.

V: conjunto de vértices de un grafo.

$|V|$: número de vértices del grafo.

V_i: i-ésimo sub-grafo.

v_i: i-ésimo vértice.

W_V: suma de pesos de los vértices del grafo G.

W_{Vsg}: suma de los pesos de los vértices del sub-grafo sg.

$w(v_i,v_j)$: peso de la arista que une los vértices v_i y v_j.

X: conjunto de variables.

x_i: i-ésima variable.

z: imagen de $f(s)$, valor de la función objetivo de la solución s.

Z: espacio objetivo.

Índice general

Índice de figuras

Índice de tablas

Summary

In real-life there exists a large number of situations where we must decide among several alternatives. Many of these situations are so easy and usual that they can be solved using a reasoning process. However, other more complex situations cannot be solved in an optimal way using rational processes, which is why formal planning is necessary about how to take accurate decisions.

In mathematics, optimization [PAR00] is the discipline which is concerned with finding inputs of a function that minimize or maximize its value, in most cases subject to constraints. Combinatorial optimization [GRO95] is a branch of optimization at the intersection of applied mathematics, computer science, and operations research. Combinatorial optimization is concerned with problems where the set of feasible solutions is discrete or can be reduced to a discrete one. Computational optimization can be defined as the process of designing and implementing algorithms for solving a large variety of optimization problems. It is now possible to solve real-life problems that in the past were thought to be unsolvable thanks to new technological developments in algorithms and computer hardware. Computational optimization includes the disciplines of operations research to model the system, mathematics to formulate the model, computer science for algorithmic design and analysis, and software engineering to implement the model.

Despite its name, optimization does not necessarily mean finding the optimum solution to a problem, since it can be unfeasible due to the characteristics of the problem. The first consideration when solving a real optimization problem is to classify it according to the computational complexity theory [GAR79]. This classification is often performed according to the most common resources, i.e. space (how much memory it takes to solve a problem), and time (how many steps it takes to solve it). Only after determining the complexity of the problem, can we decide the best way to tackle it. The relationship between the complexity classes P and NP is one of the most important open problems in theoretical

computer science and mathematics. Informally class P is the category of decision problems solvable by some algorithms within a number of steps bound by some fixed polynomial in the length of the input. On the other hand, class NP consists of all those decision problems whose positive solutions can be verified in polynomial time given the right information, or equivalently, whose solution can be found in polynomial time on a non-deterministic Turing machine [COO71]. A problem is said to be NP-hard if solving it in polynomial time would make it possible to solve all problems in class NP in polynomial time. NP-complete (non-deterministic polynomial-time complete) problems are the toughest problems in NP, in the sense that they are the ones most likely not to be in P. The reason is that if we could find a way to solve any NP-complete problem (in polynomial time), then that algorithm could be used to solve all NP problems quickly.

The increase in computational resources in recent decades has allowed the researchers to develop more complex and efficient computational algorithms. Many authors have developed heuristic methods to treat complex problems. Heuristics [GLO03a] is the art and science of discovery and invention. The word comes from the same Greek root as Eureka (*I have found it*). Heuristics are computational methods that use trial and error strategies to approximate a solution for computationally difficult problems using empirical information that has no explicit rationalization.

Heuristic techniques provide a good performance in a wide range of applications. Nevertheless, the quality of the solutions obtained can be improved by means of combining (hybridizing) two or more methods taking advantage of their particular characteristics. Some studies [TAL02] have demonstrated that, in some problems, the synergy of different heuristics outperform their individual performance.

To date, most computational optimization research has been focused on solving single-objective problems, including in some cases constraints. Nevertheless, there exists a large number of applications that require the simultaneous optimization of several objectives which are often in conflict. Consequently, some authors have proposed several multi-objective algorithms, usually based on the Pareto-dominance concept [GOL89].

It is known that the runtime required to obtain good solutions increases with the size of the problem to be solved. In some cases, the complexity is so high that even heuristic methods are not able to obtain accurate solutions in reasonable runtimes. In these cases parallel processing [BLA00] becomes an interesting, even

the best way to obtain good solutions in acceptable runtimes.

This thesis deals with the analysis, design, implementation, and evaluation of new (meta-)heuristic methods for solving real-life optimization problems. These implementations include aspects related with hybridization, multi-objective optimization, and parallel processing. The performance of these algorithms are analyzed in two test problems: the graph partitioning problem [GPA], and the optimal design of looped water distribution systems [SAV97].

The graph partitioning problem is included in the category of NP-complete problems [GAR79]. It consists of partitioning a given graph into several subgraphs, such that the number of edges connecting vertices belonging to different sub-domains is minimized, while the imbalance is maintained below a given threshold (single-objective constrained formulation) or also minimized (multi-objective formulation). The drawback of the single-objective constrained formulation is its high dependency on the imbalance constraint. An interesting way to overcome this inconvenience is to use a multi-objective formulation which considers several objectives simultaneously. It is more difficult to treat, but it seems to be more suitable for realistic applications, such as load balancing [WAL02, DEV05], VLSI design [GIL98, JIA03], task scheduling [ALE01, ERC05], etc.

On the other hand, the optimal design of water distribution networks is a real optimization problem that consists of finding the best way to convey water from the sources to the users, satisfying their requirements. Although these systems have usually been branched networks due to their lower investment costs, it has recently been demonstrated that looped water distribution networks are more suitable as their greater reliability can offset the slight increase in the cost of the closed loop networks [REC06]. It is included in the category of NP-hard problems [GUP93]. It is a non-linear, constrained, non-smooth and non-convex, and hence multi-modal problem. Although many researchers have reported algorithms for minimizing the network cost applying a large variety of techniques, such as non-linear programming [VAR97], global optimization methods [SHR98], metaheuristic approaches [CUS99, MAI03], etc., a totally satisfactory and efficient method is not available as yet. Many works have assessed the performance of these techniques using small or medium-sized benchmark networks proposed in the literature, but few of them have tested these methods with large-scale real networks. Recently, some authors have used multi-objective optimization formulations in order to consider the cost and the reliability of the network simultaneously. Reliability is concerned with the ability of the network to provide

an adequate supply to consumers, under both normal and abnormal operating conditions for a given period of time.

As both are layout problems, the main aim of the methods here proposed is to improve the quality of the solutions. Furthermore, with a view to establishing robust conclusions, several methods found in the literature have also been implemented. These methods include aspects such as single/multi-objective optimization, trajectory/population based methods, evolutionary/local-search techniques, single/hybrid alternatives, and sequential/parallel implementations. These techniques have been applied to the problems described above with a double aim: to evaluate their performance in comparison with other previous approaches, and to give new solutions to these real-life problems. The results and conclusions of this thesis help to open up new potential directions for research within the field of computational optimization.

Contributions

The main contributions of this doctoral dissertation in the field of computational optimization can be summarized in four parts, as detailed below:

Design and Implementation of Single and Multi-objective Heuristics

- rMSATS (refined Mixed Simulated Annealing and Tabu Search) [GIL02]: generalization of MSATS algorithm [GIL96] for solving partitioning problems.
- MLrMSATS (Multi-Level rMSATS) [BAÑ03]: multi-level heuristic which uses rMSATS as improvement procedure.
- SSSA (Scatter Search with Simulated Annealing) [BAÑ06d]: single-objective hybrid meta-heuristic that combines Scatter Search (SS) and Simulated Annealing (SA).
- MOSATS (Multi-objective Simulated Annealing and Tabu Search) [BAÑ06a]: multi-objective hybrid meta-heuristic that combines Simulated Annealing (SA) and Tabu Search (TS).
- MOSSSA (Scatter Search with Simulated Annealing) [BAÑ06e]: multi-objective hybrid meta-heuristic that combines Scatter Search (SS) and Simulated Annealing (SA).

Design and Implementation of Parallel Single and Multi-objective Heuristics

- PrMSATS (Parallel rMSATS) [BAÑ04a]: parallelization of rMSATS [GIL02].
- PMSATS (Parallel MLrMSATS) [BAÑ04b]: parallelization of MLrMSATS [BAÑ03].
- pPSA (Parallel Pareto Simulated Annealing) [BAÑ06b]: parallelization of PSA [CZY98].

Study, Implementation and Evaluation of Heuristics Proposed by Other Authors

- Analysis of the current state of the art in techniques for graph partitioning problem [BAÑ06f].
- Comparative study [BAÑ06c] among several local-search based multi-objective meta-heuristics found in the literature to solve partitioning problems.
- Comparative study [BAÑ06d] among several single-objective techniques based on evolutionary and local search, to design water distribution networks.
- Comparative study [BAÑ06e] among several multi-objective techniques based on evolutionary and local search, to design water distribution networks.

Treatment of Real-life Optimization Problems

- Solving partitioning problems using graphs by means of single-objective [GIL02, BAÑ03, BAÑ04a, BAÑ04b], and multi-objective methods [BAÑ06a, BAÑ06b, BAÑ06c]. These methods have been applied to several test graphs that model circuits and networks.
- Optimizing the design of water distribution networks using single-objective [BAÑ06d] and multi-objective [BAÑ06e] methods. These methods have been applied to optimize three looped networks, including an irrigation one.

Structure of the Thesis

The structure of this thesis is as follows.

- Chapter 1 describes general optimization concepts. It also provides information about the main single-objective meta-heuristics found in the literature.
- Chapter 2 offers general information about multi-objective optimization concepts. In addition, the most important multi-objective meta-heuristics found in the literature are briefly described.
- Chapter 3 introduces the optimization problems treated in this dissertation. The first one is the graph partitioning problem [GPA], which is a combinatorial optimization problem of practical interest in many engineering fields. In addition to the classic single-objective formulation of this problem, a Pareto-based multi-objective formulation is also described. The second problem is the optimal design of looped water distribution networks [SAV97] using two different formulations. On the other hand, a constrained single-objective formulation which tries to minimize the network investment cost, while some constraints are satisfied. On the other hand, a multi-objective formulation that tries to minimize the network investment cost, while maximizing its reliability.
- Chapter 4 presents three single-objective hybrid algorithms. These algorithms are compared and evaluated with other methods found in the literature in the test problems described in Chapter 3.
- Chapter 5 presents two multi-objective hybrid algorithms. These problems are evaluated using multi-objective formulations of the test problems described in Chapter 3. The results obtained in both problems are also compared with adaptations of methods previously proposed by other authors.
- Chapter 6 analyzes the advantages of parallel processing when solving high complexity problems using single/multi-objective methods. Parallel versions of some methods described in Chapters 4 and 5 are presented.
- Finally, Chapter 7 presents the conclusions of this thesis, the contributions of the proposed heuristic methods to the computational optimization research field, and the contributions to the problems tackled.

Prólogo

En la vida real se dan una gran cantidad de situaciones en las cuales el ser humano se encuentra en la disyuntiva de tener que elegir entre diferentes alternativas. Muchas de esas situaciones son habituales y escasamente complejas, por lo que se pueden resolver directamente siguiendo procesos racionales relativamente sencillos. Sin embargo, existen otros problemas de mayor complejidad que no se pueden resolver de forma óptima siguiendo dichos procesos racionales, por lo que es necesario llevar a cabo una planificación formal para tomar la decisión óptima.

En matemáticas, la optimización [PAR00] es la disciplina encargada de encontrar entradas a una función que minimice o maximice su valor, en muchos casos sujetas a restricciones. La optimización combinatoria [GRO95] es una rama de la optimización con aspectos comunes de matemática aplicada, ciencias de la computación e investigación operativa que trata problemas cuyas soluciones son discretas. La optimización combinatoria puede definirse como el proceso de diseñar e implementar algoritmos para resolver una gran variedad de problemas de optimización. En la actualidad es posible resolver problemas de la vida real que en el pasado eran intratables gracias a los avances tecnológicos en algorítmica y en hardware de computación. La optimización computacional incluye las disciplinas de investigación operativa para modelar el sistema, matemáticas para formular el modelo, ciencias de la computación para el diseño y análisis de algoritmos, e ingeniería del software para implementar el modelo.

Una de las cuestiones importantes que hay que considerar a la hora de resolver un determinado problema de optimización es clasificarlo en función de la teoría de la complejidad algorítmica [GAR79]. Esta clasificación se rige principalmente por criterios de espacio (cuánta memoria se necesita para resolver un problema) y tiempo (cuántos pasos se requieren para resolverlo). Sólo tras realizar dicha clasificación puede determinarse la mejor forma de resolver el problema. La relación entre la clase de problemas P (polinómico) y la clase NP (no

polinómico) es una de las más utilizadas en computación y matemáticas. Informalmente, la clase P está formada por los problemas de decisión que pueden ser resueltos mediante algunos algoritmos en un número de pasos acotado por un polinomio fijo en función del tamaño de la entrada. Por otro lado, la clase de problemas NP incluye a todos los problemas de decisión cuyas soluciones pueden ser verificadas en un tiempo polinómico dada la información correcta, o lo que es lo mismo, pueden ser encontradas en tiempo polinómico haciendo uso de una máquina de Turing no determinista [COO71]. Un problema es NP-duro (en inglés non-deterministic polynomial-time hard, NP-hard) si al resolverlo en un tiempo polinómico permitiera resolver todos los problemas de la clase NP en un tiempo polinómico. Los problemas NP-completos (en inglés NP-complete) son los problemas NP más difíciles, en el sentido de que es menos probable que pertenezcan a la clase P, ya que si se pudiera encontrar la forma de resolver cualquier problema NP-completo rápidamente (en un tiempo polinómico), entonces sería posible utilizar ese algoritmo para resolver todos los problemas NP rápidamente.

La complejidad de estos problemas depende de los parámetros de entrada, usualmente del tamaño del problema a resolver, por lo que cuando dicho tamaño aumenta, el problema se vuelve rápidamente inabordable. Esto provoca que la aplicación de técnicas deterministas no sea eficiente para instancias de problemas de tamaño elevado. El incremento de los recursos de cómputo experimentado en las últimas décadas ha permitido que los problemas de la vida real que previamente no eran abordados lo sean en la actualidad. Por esta razón, en los últimos años hemos podido observar un incremento gradual en el número y sofisticación de algoritmos computacionales.

Con el objetivo de obtener soluciones aproximadas para este tipo de problemas, en las últimas décadas numerosos investigadores han desarrollado técnicas heurísticas [GLO03a]. La palabra viene de la misma rama que la palabra griega Eureka que significa *lo he encontrado*, y se puede definir de forma genérica como el arte y ciencia de descubrir e inventar. Las heurísticas son métodos o algoritmos exploratorios para la resolución de problemas en el que las soluciones se obtienen por la evaluación del progreso logrado en la búsqueda de un resultado final [GLO03a]. Las técnicas heurísticas no aseguran soluciones óptimas, sino solamente soluciones válidas, aproximadas, y frecuentemente no es posible justificar en términos estrictamente lógicos la validez del resultado.

De forma general, estas técnicas ofrecen un buen rendimiento en un amplio número de aplicaciones. Sin embargo, la calidad de las soluciones obtenidas

puede ser mejorada mediante una adecuada combinación (hibridación) de diferentes métodos que permitan aprovechar las ventajas que ofrecen dichas técnicas de forma separada. Existen importantes estudios [TAL02] que describen cómo la sinergia de diferentes heurísticas permite mejorar la calidad de las soluciones en determinados problemas. Cuando una técnica heurística se generaliza de forma que puede ser aplicada a cualquier tipo de problema recibe el nombre de meta-heurística.

Hasta hace relativamente poco tiempo, la inmensa mayoría de problemas han sido modelados mediante una formulación mono-objetivo, incluyendo o no restricciones adicionales. Sin embargo, existe un elevado número de aplicaciones que requieren de la optimización simultánea de varios objetivos, que suelen estar en conflicto [DEB01]. Por esta razón, diversos autores han propuesto durante los últimos años diferentes procedimientos de optimización multi-objetivo, usualmente basados en el concepto de óptimo de Pareto [GOL89].

Tanto para el caso de tratar problemas mono-objetivo como multi-objetivo, y pese a las mejoras computacionales que han tenido las estaciones de trabajo, existen problemas de tal complejidad e instancias de problemas tan grandes que los métodos heurísticos pueden resultar inservibles debido a que requieran tiempos de ejecución muy elevados para encontrar soluciones aceptables. Por este motivo, el uso del procesamiento paralelo [BLA00] es probablemente la herramienta más adecuada para acotar dichos tiempos de respuesta sin deteriorar la calidad de las soluciones obtenidas, o para mejorar la calidad de las soluciones alcanzadas en los límites de tiempo establecidos.

En esta tesis doctoral se analizan dos problemas de complejidad NP: la repartición de grafos [GPA] y el diseño óptimo de redes de distribución de agua [SAV97]. El problema de repartición de grafos está incluido en la categoría de problemas NP-completos [GAR79] y consiste en dividir un grafo en varios subgrafos de forma que el número de aristas que conectan vértices pertenecientes a diferentes sub-grafos sea mínimo, mientras que el desequilibrio entre la suma de pesos de cada sub-grafo se mantiene por debajo de un umbral predeterminado (formulación mono-objetivo con restricciones), o también se minimiza (formulación multi-objetivo). El principal inconveniente de la formulación mono-objetivo radica en la alta dependencia de la restricción de desequilibrio. Aunque la formulación multi-objetivo es más difícil de abordar, resulta más adecuada para aplicaciones realistas [DEB01]. Algunas aplicaciones reales donde se aplica la repartición de grafos son la distribución equilibrada de carga en computadores [WAL02, DEV05], el diseño VLSI [GIL98, JIA03], la planificación de tareas

[ALE01, ERC05], etc. Por otro lado, el problema del diseño de redes de distribución de agua consiste en encontrar la mejor forma de suministrar agua a los usuarios desde las fuentes, a la vez que se satisfacen sus requisitos. Aunque, debido a su bajo coste de implantación esos sistemas han sido tradicionalmente redes ramificadas, en la actualidad es una opinión ampliamente extendida entre los investigadores que las redes malladas constituyen una alternativa más eficiente ya que el incremento en el coste respecto a las ramificadas se ve claramente compensado por su mayor fiabilidad [REC06]. Dicha fiabilidad hace referencia a la capacidad de la red de proveer un suministro adecuado a los consumidores bajo condiciones de operación normales y anormales en un periodo de tiempo dado. Aunque son muchos los investigadores que han desarrollado técnicas de optimización para resolver este problema, incluyendo la programación no lineal [VAR97], la optimización global [SHR98], las aproximaciones heurísticas [CUS99, MAI03], etc., no existe hasta la actualidad ningún método que resuelva dicho problema de forma totalmente satisfactorio. Diferentes trabajos han tratado el rendimiento de estas técnicas en redes de prueba de tamaño pequeño y mediano [ALP77], pero pocos de ellos han trabajado con redes de prueba de gran escala. Recientemente algunos autores han utilizado formulaciones multi-objetivo con el fin de considerar el coste de diseño y la fiabilidad de la red de forma simultánea [TOD00, FAR03]. Dicha fiabilidad viene determinada por la capacidad de la red para suministrar durante un periodo de tiempo dado un adecuado suministro a los consumidores bajo condiciones de operación normales o anormales.

El objetivo principal de esta tesis doctoral es el diseño e implementación de diferentes técnicas heurísticas simples e híbridas, mono-objetivo y multi-objetivo, secuenciales y paralelas, para solucionar problemas de optimización relevantes, tanto desde el punto de vista de sus implicaciones teóricas como desde la perspectiva de su incidencia socio-económica. Esos problemas son la repartición de grafos, y la optimización del diseño de redes de distribución de agua. Dado que ambos se pueden considerar como problemas de diseño o *layout*, la principal función de los algoritmos aquí propuestos es mejorar la calidad de las soluciones obtenidas, aunque también se tendrán en cuenta otros aspectos, como los relacionados con el manejo de restricciones, el coste computacional requerido para su resolución, etc. Además de los algoritmos propuestos, y con el objetivo de establecer comparaciones robustas entre técnicas, se han adaptado a ambos problemas diferentes técnicas encontradas en la literatura y propuestas previamente por otros autores. Los resultados y conclusiones obtenidos en esta

tesis contribuyen a abrir nuevas áreas potenciales de investigación dentro del campo de la optimización computacional.

Alcance y Contribuciones

Las principales contribuciones de esta tesis al campo de la optimización computacional se pueden resumir en tres grandes bloques, tal y como se describe a continuación:

Diseño e Implementación de Heurísticas Mono-objetivo/Multi-objetivo

- rMSATS (refined Mixed Simulated Annealing and Tabu Search) [GIL02]: generalización del algoritmo MSATS [GIL96] para resolver problemas de repartición utilizando la formulación mono-objetivo.
- MLrMSATS (Multi-Level rMSATS) [BAÑ03]: extensión de rMSATS haciendo uso del paradigma multinivel.
- SSSA (Scatter Search with Simulated Annealing) [BAÑ06d]: meta-heurística híbrida mono-objetivo que combina Búsqueda Dispersa (SS) y Enfriamiento Simulado (SA).
- MOSATS (Multi-objective Simulated Annealing and Tabu Search) [BAÑ06a]: meta-heurística híbrida multi-objetivo que combina Enfriamiento Simulado (SA) y Búsqueda Tabú (TS).
- MOSSSA (Multi-objective Scatter Search with Simulated Annealing) [BAÑ06e]: meta-heurística híbrida multi-objetivo que combina Búsqueda Dispersa (SS) y Enfriamiento Simulado (SA).

Diseño e Implementación de Heurísticas Paralelas Mono-objetivo y Multi-objetivo

- PrMSATS (Parallel rMSATS) [BAÑ04a]: Paralelización de rMSATS mediante el uso de diferentes esquemas de enfriamiento.
- PMSATS (Parallel MLrMSATS) [BAÑ04b]: Paralelización de MLrMSATS.
- pPSA (Parallel Pareto Simulated Annealing) [BAÑ06b]: Paralelización de PSA [CZY98].

Estudio, Implementación y Evaluación de Heurísticas Propuestas por Otros Autores

- Estudio del estado del arte en algoritmos para resolver el problema de repartición de grafos [BAÑ06f].
- Estudio comparativo [BAÑ06c] entre diferentes técnicas multi-objetivo basadas en búsqueda local propuestas previamente por otros autores para resolver el problema de repartición de grafos.
- Estudio comparativo [BAÑ06d] entre diferentes técnicas mono-objetivo, tanto evolutivas como basadas en búsqueda local en el problema de redes de distribución de agua.
- Estudio comparativo [BAÑ06e] entre diferentes técnicas multi-objetivo basadas en computación evolutiva y búsqueda local en el problema de diseño de redes de distribución de agua.

Resolución de Problemas de Optimización del Mundo Real

- Optimización del problema de repartición de grafos mediante técnicas mono-objetivo [GIL02, BAÑ03, BAÑ04a, BAÑ04b] y multi-objetivo [BAÑ06a, BAÑ06b, BAÑ06c], y comparativa con métodos encontrados en la literatura. Estas técnicas han permitido obtener los mejores resultados obtenidos hasta la actualidad en diferentes grafos de prueba [GPA].
- Optimización de redes de distribución de agua mediante técnicas mono-objetivo [BAÑ06d], y multi-objetivo [BAÑ06e]. Los resultados obtenidos en ambas formulaciones son de gran calidad, alcanzando en muchos algunos casos los mejores resultados conocidos hasta el momento.

Estructura General de la Tesis

A continuación se ofrece una breve descripción de la estructura de esta tesis haciendo referencia al contenido de los capítulos incluidos en ella.

- El Capítulo 1 incluye los conceptos básicos de optimización, así como las principales técnicas meta-heurísticas encontradas en la literatura, y su uso en problemas mono-objetivo.
- El Capítulo 2 se centra en la optimización multi-objetivo. Al igual que en el caso mono-objetivo, se ofrece una descripción general de este tipo de

problemas, así como los principales métodos existentes para su resolución.
• El Capítulo 3 describe los problemas reales abordados en esta tesis. Por un lado se describe el problema de la repartición de grafos, que resulta de gran interés en diferentes campos de la ingenería. Además de resolver la formulación clásica de este problema, también se trata una versión multi-objetivo que considera la optimización de varios objetivos simultáneamente. Por otro lado se ofrece una descripción del problema del diseño óptimo de redes malladas de distribución de agua. En concreto se analizan dos formulaciones diferentes: una formulación mono-objetivo con restricciones que trata de minimizar el coste de diseño de la red a la vez que se satisfacen ciertas restricciones de presión en los nodos de demanda, y por otro lado una formulación multi-objetivo que minimiza el coste de diseño a la vez que se maximiza la fiabilidad del mismo.
• El Capítulo 4 presenta tres nuevos algoritmos híbridos mono-objetivo. Todos ellos comparados con otras técnicas encontradas en la literatura haciendo uso de los problemas de test descritos en el Capítulo 3.
• El Capítulo 5 presenta dos nuevas meta-heurísticas híbridas multi-objetivo. Al igual que para los algoritmos multi-objetivo, se evaluan sus rendimientos en términos de calidad frente a otras técnicas existentes en la literatura. Para ello se hace uso de las formulaciones multi-objetivo de los problemas de prueba descritos en el Capítulo 3.
• El Capítulo 6 está enfocado a analizar las ventajas del uso del procesamiento paralelo a la hora de mejorar la calidad de las soluciones en problemas de optimización mono-objetivo y multi-objetivo con meta-heurísticas. Se han paralelizado algunos de los algoritmos propuestos en los capítulos 4 y 5, así como diferentes técnicas propuestas por otros autores.
• Finalmente, el Capítulo 7 ofrece las conclusiones de esta tesis para cada uno de los aspectos analizados, esto es, las aportaciones que proporcionan los procedimientos heurísticos propuestos, la paralelización de algunos de ellos, así como las aportaciones realizadas en los problemas de optimización tratados.

Capítulo 1

Introducción a la Optimización

En este capítulo se introduce el concepto general de optimización, y se ofrece una visión general de las principales técnicas heurísticas encontradas en la literatura para abordar problemas de optimización de gran complejidad. En concreto, se revisan los distintos tipos de heurísticas de optimización, clasificandolas en función de si trabajan con una sola solución (métodos basados en trayectorias) o utilizan poblaciones de soluciones (métodos poblacionales). Algunos de los métodos descritos en esta sección serán utilizados en capítulos posteriores como métodos de referencia parar evaluar la calidad de los procedimientos propuestos en esta tesis doctoral.

1.1. Conceptos Generales de Optimización

Existen áreas del conocimiento y del quehacer humano donde surgen problemas relacionados con la mejora de determinados procesos o la obtención de mejores soluciones (generales o particulares). La disciplina que estudia este tipo de problemas y sus respectivas alternativas es conocida como optimización [PAR00]. Los problemas de optimización pueden clasificarse en función de diferentes factores como son su complejidad, la existencia o no de restricciones, su carácter estático o dinámico, lineal o no lineal, mono-objetivo o multi-objetivo, etc. En cuanto a las técnicas de búsqueda, estas se pueden clasificar en función de si aseguran obtener el resultado óptimo (técnicas exactas) o si por el contrario permiten obtener soluciones cercanas al óptimo (técnicas aproximadas) [PUC05].

La optimización combinatoria [GRO95] consiste en encontrar la mejor solución (óptima) de entre un conjunto finito de soluciones alternativas. La calidad u optimalidad de las soluciones vendrá definida por la capacidad de dichas soluciones para minimizar o maximizar una determinada función, denominada función objetivo, compuesta por un conjunto determinado de variables definidas sobre un conjunto discreto. Esta disciplina tiene numerosas aplicaciones en ámbitos como las ciencias, la ingeniería, la industria, la logística, la administración de organizaciones, etc. Como ejemplos podemos mencionar, entre otros, el diseño VLSI [SRV05], el diseño de redes de telecomunicación [KEL98], la asignación de tareas a procesadores [RAS03], el enrutamiento y carga de vehículos en redes de distribución [GLO01], la planificación de la producción [KUB02], la selección de carteras financieras [ROC00], la planificación de la generación de electricidad [LIN00], el diseño de proteinas [GOR99], la distribución de ambulancias en una región para asegurar un cierto nivel de servicio a su población [GEN01], etc. La gran variedad de aplicaciones que requieren la ayuda de optimizadores ha provocado un gran interés en las últimas décadas en el desarrollo de nuevos métodos.

Siguiendo la definición utilizada por Blum y Roli [BLU03], un problema de optimización combinatoria (en inglés Combinatorial Optimization Problem, COP) se puede definir como COP=(F,f), utilizando los siguientes conceptos:

- un conjunto de variables $X=\{x_1 , ..., x_n\}$;
- dominios de las variables $D_1 , ..., D_n$;
- una función objetivo $f : D_1$ x ... x $D_n \rightarrow \Re^+$;
- un conjunto de restricciones entre las variables;

El conjunto de todas las soluciones que satisfacen las restricciones entre las variables constituye el llamado *espacio de búsqueda*, o *espacio de soluciones*, F. Formalmente, F={s={(x_1,u_1),...,(x_n,u_n)} con $u_i \in D_i$, i=1,...,n, tal que s satisface todas las restricciones del problema}.

La función objetivo transforma los vectores de decisión (s) desde el espacio de búsqueda hacia un espacio objetivo $Z \in \Re$. De esta forma, dado un problema de minimización, el objetivo es encontrar aquella solución $s' \in F$ tal que $f(s') \leq f(s)$, $\forall s \in F$. La solución s' se denomina óptimo global del COP. La Figura 1.1(a) muestra gráficamente un problema de optimización con varios mínimos locales, y un mínimo global correspondiente a la solución óptima del problema.

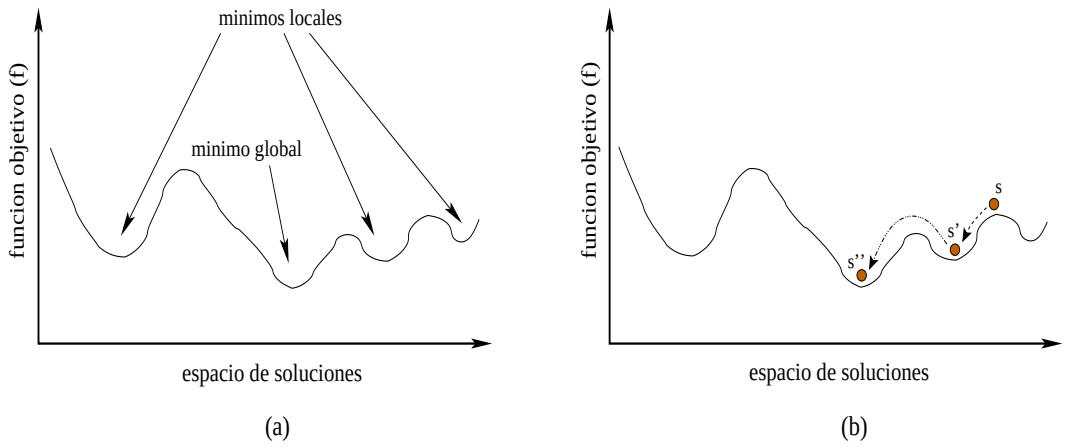

Figura 1.1: (a) Problema de optimización con varios mínimos locales y un mínimo global, (b) mejora de soluciones.

El estudio de los problemas de optimización combinatoria [GRO95] es un tema en el que desde hace tiempo se está invirtiendo bastante esfuerzo investigador. En optimización combinatoria confluyen la matemática discreta, la teoría de algoritmos, y la programación lineal. Todo problema de *programación lineal* consiste en una función objetivo lineal que se ha de minimizar o maximizar a la vez que sus variables están sujetas a restricciones también lineales. Si además se exige que las variables tomen valores enteros nos encontramos ante un problema de *programación lineal-entera*. Un amplio número de problemas de optimización combinatoria pueden ser resueltos en tiempo polinómico utilizando los métodos

y teoría de la programación lineal. Sin embargo, en el caso de que el problema a resolver pertenezca a la categoría de los llamados NP-completos [GAR79], la capacidad de encontrar una solución exacta es inversamente proporcional al tamaño del problema. Bajo dichas circunstancias, el uso de técnicas deterministas suele resultar ineficiente, por lo que resulta conveniente aplicar técnicas heurísticas [GLO03a] que permitan encontrar soluciones, si no óptimas, al menos aproximadas en tiempos de ejecución aceptables.

1.2. Meta-heurísticas de Optimización

En las últimas décadas se ha desarrollado una nueva clase de algoritmos de aproximación que básicamente tratan de combinar métodos heurísticos básicos en un marco de alto nivel orientado a buscar la eficiencia en el proceso de búsqueda. Durante los años 80 y principios de los 90 estas técnicas se conocieron como heurísticas modernas [GLO93]. Dichas técnicas consisten en procedimientos sistemáticos de prueba que ofrecen soluciones aceptables, no necesariamente óptimos absolutos, para problemas donde el espacio de soluciones es indeterminado o lo suficientemente amplio como para que no pueda ser evaluado en un tiempo aceptable. En muchos casos, las técnicas heurísticas se diseñan en función de las características particulares del problema a resolver. En la actualidad existe un gran número de técnicas heurísticas de optimización [GLO03a].

Además del concepto de heurística, es habitual hablar de meta-heurística. Una meta-heurística se puede describir como un proceso iterativo maestro que guía y modifica las operaciones de heurísticas subordinadas para producir de forma eficiente soluciones de alta calidad. Las heurísticas subordinadas pueden ser procedimientos de alto o bajo nivel, un método de búsqueda local o simplemente un método constructivo [VOB99]. Aunque se suele considerar como meta-heurística a toda aquella generalización de una determinada heurística para cualquier tipo de problema, en la práctica es necesario analizar detalladamente el problema a resolver para determinar cuál de ellas es la que se presupone puede tener un mayor éxito en la búsqueda de soluciones, aunque no se pueda determinar con total seguridad si alcanzaremos dicho éxito.

A continuación describimos las características básicas de las meta-heurísticas, según la descripción ofrecida por Blum y col. [BLU03].

- son estrategias que guían el proceso de búsqueda;
- exploran eficientemente el espacio de búsqueda con el objetivo de encon-

trar soluciones próximas al óptimo global;
• las técnicas que constituyen las meta-heurísticas varían entre métodos de búsqueda local simple a complejos métodos de aprendizaje;
• pueden incorporar mecanismos para evitar quedar atrapado en óptimos locales del espacio de búsqueda;
• permiten un nivel de descripción abstracto, no específico del problema;
• pueden hacer uso de conocimiento del dominio específico de forma que las heurísticas son controladas por una estrategia de nivel superior;
• las meta-heurísticas avanzadas utilizan la experiencia de búsqueda (haciendo uso de alguna forma de memoria) para guiar la búsqueda.

Entre las principales técnicas meta-heurísticas de optimización [GLO03a] podemos encontrar el enfriamiento simulado (en inglés Simulated Annealing, SA) [KIR83], la búsqueda tabú (en inglés Tabu Search, TS) [GLO93, GLO97], los algoritmos genéticos (en inglés Genetic Algorithms, GA) [HOL75, GOL89], la búsqueda dispersa (en inglés Scatter Search, SS) [GLO77, MAR06], los algoritmos meméticos (en inglés Memetic Algorithms, MA) [MOS00], la búsqueda en vecindario variable (en inglés Variable Neighborhood Search, VNS) [HAN99], la búsqueda local guiada (en inglés Guied Local Search, GLS) [VOU98], los procedimientos de búsqueda basados en procedimientos adaptativos aleatorizados avaros (en inglés Greedy Randomized Adaptive Search Procedures, GRASP) [FEO95], la optimización mediante colonias de hormigas (en inglés Ant Colony Optimization, ACO) [DOR99], la búsqueda local iterada (en inglés Iterated Local Search, ILS) [RAM02], las redes neuronales (en inglés Neural Networks, NN) [SMI99], etc. Además, se pueden incluir las técnicas híbridas que combinan aspectos de diferentes meta-heurísticas [TAL02], así como versiones paralelas de dichos métodos [CAH04].

En función de las características propias de cada meta-heurística se pueden establecer diferentes clasificaciones. A continuación ofrecemos una de las clasificaciones más aceptadas en la actualidad [BLU03].

• Técnicas *inspiradas* vs. *no inspiradas en la naturaleza*. Muchas meta-heurísticas, como por ejemplo los algoritmos genéticos [HOL75], enfriamiento simulado [KIR83], optimización mediante colonias de hormigas [DOR99], etc. basan su funcionamiento en aspectos inspirados de la naturaleza. Otras, como por ejemplo búsqueda tabú [GLO93] o la búsqueda local guiada [VOU98], no simulan ningún aspecto de la naturaleza.

• Técnicas *basadas en trayectorias* vs. *poblacionales*. Un aspecto diferenciador entre meta-heurísticas radica en el número de soluciones que se utilizan en el proceso de optimización. Meta-heurísticas como enfriamiento simulado [KIR83], o búsqueda tabú [GLO93] utilizan una sola solución durante el proceso de búsqueda, por lo que se suelen denominar métodos de trayectoria ya que la solución describe una trayectoria desde la solución de partida hasta encontrar la solución final. Por otro lado, técnicas como los algoritmos genéticos [HOL75] hacen uso de un conjunto de soluciones (población) que son optimizadas de forma simultánea durante la búsqueda. Normalmente los métodos poblacionales suelen conseguir soluciones de más calidad debido a la ventaja que supone explorar simultáneamente diferentes áreas del espacio de búsqueda.

• Técnicas *estáticas* vs. *dinámicas.* Otro aspecto diferenciador entre meta-heurísticas radica en la función de aptitud utilizada. Mientras que en la mayoría de heurísticas utilizan la misma función objetivo durante todo el proceso de búsqueda otras, como la búsqueda local guiada [VOU98], modifican dicha función en tiempo de ejecución, lo que ayuda a escapar de mínimos locales.

• Técnicas basadas en estructuras de *vecindario único* vs. *vecindarios múltiples.* La mayoría de los algoritmos trabajan con una estructura de vecindario simple, es decir, la topología del espacio objetivo no cambia durante la búsqueda. Sin embargo, existen meta-heurísticas como la búsqueda en vecindario variable [HAN99] que diversifican la búsqueda mediante el uso de diferentes espacios de soluciones.

• Meta-heurísticas *con memoria* vs. *sin memoria.* Uno de los criterios más utilizados para clasificar las meta-heurísticas es el uso que hacen de su historia de búsqueda, es decir, si utilizan memoria o no. Heurísticas como el enfriamiento simulado [KIR83] utilizan exclusivamente el estado actual del proceso de búsqueda a la hora de determinar la próxima forma de actuación. Otras técnicas como la búsqueda tabú [GLO93] utilizan información previa (memoria) del proceso de búsqueda a la hora de tomar nuevas decisiones. Existen dos formas principales de hacer uso de la memoria. Una de ellas consiste en incorporar memoria a corto plazo teniendo en cuenta las últimas decisiones tomadas (movimientos realizados, soluciones visitadas, etc.). En otros casos se hace uso de memoria a largo plazo, que considera toda o una gran parte de la búsqueda realizada.

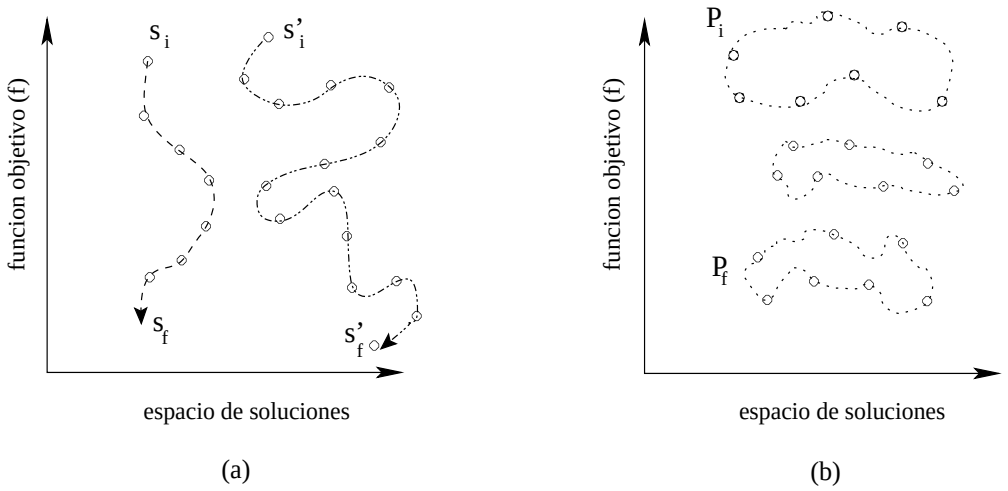

Figura 1.2: (a) Meta-heurísticas basadas en trayectorias, (b) poblacionales.

1.2.1. Técnicas Meta-heurísticas Basadas en Trayectorias

Las llamadas meta-heurísticas basadas en trayectorias vienen caracterizadas, como su propio nombre indica, por el hecho de que se utiliza una única solución durante el proceso de búsqueda que describe una trayectoria en el espacio objetivo durante dicho proceso. Todos estos métodos tienen en común que parten de una solución inicial que es modificada aplicando determinadas estrategias particulares. A continuación se describen brevemente algunas de las meta-heurísticas de trayectorias más conocidas.

Ascenso de Colinas (HC)

El método básico de búsqueda local es el conocido como ascenso de colinas (en inglés Hill Climbling, HC) [JAC04]. Dicho nombre, que para problemas de minimización se denomina descenso de colinas, se debe a que cada para admitir una determinada solución, esta debe conllevar una mejora respecto a la anterior, es decir, no se permiten movimientos que empeoren la calidad de la solución. Por esta razón, el algoritmo de ascenso de colinas se detiene cuando se encuentra un óptimo local, no siendo efectivo en problemas combinatorios complejos. Así, por ejemplo, dada la solución s de la Figura 1.1(b), HC solamente es capaz de alcanzar el mínimo local representado por la solución s', y no el mínimo global (s"). Por esta razón, esta técnica ha quedado relegada por otras que permiten

escapar de dichos óptimos locales. El comportamiento de la técnica de ascenso de colinas corresponde a la trayectoria descrita por la solución s en la Figura 1.2(a). El Algoritmo 1.2.1 describe este procedimiento en alto nivel.

Algoritmo 1.2.1

Begin HC
  $s \leftarrow$ GenerarSoluciónInicial();
  Hacer
    $s'\leftarrow$Obtener_Solución_Vecina($N(s)$);
    Si ($f(s')<f(s)$) **entonces**
      $s\leftarrow s'$;
  Mientras (exista mejora posible);
  Retornar s;
End HC

Enfriamento Simulado (SA)

El enfriamiento simulado (en inglés Simulated Annealing, SA) [KIR83] es una técnica de búsqueda local propuesta con el objetivo de evitar, en la medida de lo posible, quedar atrapados en óptimos locales tal y como sucede en los métodos de ascenso de colinas. El funcionamiento de SA es similar al proceso de enfriamiento al que es sometido un metal fundido. Cuando el metal se funde, sus partículas toman configuraciones aleatorias siendo propenso a deformarse. Tras la fundición, su temperatura disminuye lentamente hasta alcanzar un estado de mínima energía, proceso durante el cual dichas partículas van tomando una configuración robusta, y por tanto se dificulta su deformación. La analogía en el campo computacional radica en que la técnica de enfriamiento simulado trabaja con una solución inicial aleatoria, que se va optimizando de forma que al inicio del proceso se permiten variaciones de las soluciones que empeoren la función de aptitud con el propósito de escapar de los óptimos locales, mientras que conforme el proceso avanza disminuye la probabilidad de aceptar soluciones que empeoren dicha función. Esta analogía ha sido probada con éxito en numerosos problemas de optimización [JOH89, CUS99], mostrando gran habilidad para evitar quedar atrapado en óptimos locales. Este hecho, junto a la sencillez de su implementación, hizo que se popularizara a partir de la década de los ochenta.

El primer aspecto a determinar a la hora de ejecutar SA es establecer los parámetros que determinan el plan de enfriamiento. Estos parámetros son el valor inicial (T_i) del parámetro de control, llamado *temperatura* (t), su coeficiente de enfriamiento (T_{enfr}), el número de nuevas soluciones a generar antes de decrementar la temperatura, así como la temperatura de parada (T_{parada}). La forma usual de reducir la temperatura consiste en multiplicar T_{enfr} por el valor actual de la temperatura en la iteración actual, t. Estos parámetros se deben ajustar en función del tipo y tamaño del problema a resolver. Tras esto, SA obtiene una solución inicial, s y a partir de esta se genera una nueva solución vecina perteneciente al vecindario de s ($N(s)$). Se calcula la diferencia de valor de aptitud entre ambas soluciones, determinando posteriormente si se acepta o no la nueva solución candidata. Esta decisión se suele tomar haciendo uso del Criterio de Metrópolis [MET53]. Este criterio probabilista, cuyo uso en problemas de optimización combinatoria se remonta a los años ochenta, se basa en aceptar nuevas soluciones en función de la mejora o empeoramiento en la calidad de la solución vecina con respecto a la original, y en función del valor actual de t.

Algoritmo 1.2.2

Begin SA (T_i, T_{enfr}, T_{parada})
  s←Generar_Solución_Inicial();
  t← T_i;
  Hacer
    s'←Obtener_Solucion_Vecina($N(s)$);
    aceptar←Metropolis (s,s',t);
    Si (aceptar=TRUE) **entonces**
      s←s';
    t←t ·T_{enfr};
  Mientras (t>T_{parada});
  Retornar s;
End SA

SA comienza con la inicialización de la temperatura t=T_i a un valor elevado, lo cual proporciona una probabilidad también alta de aceptar un movimiento que empeore el valor de la función de aptitud, con la aspiración de escapar de

óptimos locales. En cada iteración se va reduciendo la temperatura, y con ello dicha probabilidad, de forma que inicialmente se realiza una diversificación de la búsqueda sin controlar demasiado el coste de las soluciones visitadas, mientras que conforme avanza el proceso se va intensificando la búsqueda en un área determinada del espacio objetivo. Para comprender mejor las ventajas de utilizar SA, podemos analizar la Figura 1.1(b). Mientras que utilizando HC la solución inicial s sólo puede alcanzar la posición s', haciendo uso de SA es posible alcanzar el óptimo global, representado por la solución s". Del mismo modo, la solución s' de la Figura 1.2(a) puede representar una posible trayectoria de una solución al aplicar SA. Como se observa, mientras que en HC no es posible empeorar la función de coste, en SA si se permite, lo que ayuda a escapar de optimos locales. El pseudocódigo general de SA se describe en el Algoritmo 1.2.2.

Existe una demostración matemática de la convergencia del SA hacia el óptimo global de un problema si la temperatura desciende de forma infinitamente lenta [DEK91]. Obviamente, esto no se puede implementar en una aplicación real en la que el procedimiento debe concluir en un tiempo finito. Por tanto, la elección de los valores de T_i, T_{enfr}, T_{parada} constituye un factor primordial para el éxito del enfriamiento simulado en un determinado problema. Trabajos anteriores han demostrado que el uso de valores próximos a la unidad de T_{enfr} ($0.9<T_{enfr}<1$) obtienen un buen rendimiento [NOI98]. Existen diferentes estrategias para establecer el número de iteraciones que se realizan con cada temperatura. Una posible estrategia es la de aplicar varias iteraciones con un número medio-bajo de temperaturas, que se determina en función del tamaño de la vecindad o del espacio de soluciones. Otra opción consiste en realizar pocas iteraciones, incluso a veces una sola, para muchas temperaturas distintas a cambio de decrementar la temperatura muy lentamente [NOI98].

En lo que hay total unanimidad es en el hecho de que resulta mucho más interesante invertir más tiempo en temperaturas bajas para asegurar que el óptimo local ha sido totalmente explorado [NOI98]. En teoría, la temperatura debería disminuir hasta cero, pero sin embargo no hay necesidad de disminuirla tanto, sino que es suficiente con mantener un criterio de parada próximo a cero (ej: $T_{parada}\approx 0.01$). No obstante, la experiencia adquirida en cada problema será la que determine el enfoque más conveniente a tomar.

Búsqueda Tabu (TS)

La búsqueda tabú (en inglés Tabu Search, TS) tiene sus orígenes en diversos trabajos publicados a principios de la década de los ochenta, aunque fue presentada formalmente unos años después por Fred Glover y col. [GLO93]. Esta meta-heurística trata de encontrar el óptimo del problema a resolver, combinando la búsqueda local con una heurística para evitar quedar atrapado en óptimos locales y evitar entrar en ciclos. A tal efecto, hace uso del concepto de memoria y lo implementa mediante estructuras simples con el objetivo de dirigir la búsqueda teniendo en cuenta la historia previa. Puede decirse que se lleva a cabo un cierto aprendizaje y que la búsqueda es inteligente. El principio fundamental que rige a los procedimientos de TS se basa en que es mejor una mala decisión basada en información que una buena decisión al azar, ya que, en un sistema que emplea memoria, una mala elección basada en una estrategia proporcionaría claves útiles para continuar la búsqueda de forma eficiente. Una buena elección fruto del azar no proporcionaría ninguna información para posteriores acciones. La forma de proceder de TS consiste en ir pasando de una determinada solución s a otras en el entorno de la primera, $N(s)$. Así pues, cada solución $s \in S$ tiene asociado un conjunto de vecinos, $N(s) \in S$, denominado vecindad de s. Cada solución $s' \in N(s)$ puede alcanzarse directamente a partir de s mediante una operación llamada *movimiento*.

Normalmente, al utilizar TS las vecindades se suponen simétricas, es decir, s' es un vecino de s si y sólo si s es un vecino de s'. Sin embargo, en lugar de considerar todo el entorno de s, TS define el entorno reducido N^*(s) como aquellas soluciones disponibles del entorno de s que son alcanzables directamente. Existen muchas formas de definir el entorno reducido de una solución ($N^*(s)$). La más sencilla, llamada *memoria a corto plazo*, consiste en etiquetar como tabú las soluciones previamente visitadas en un pasado cercano. Así, en una iteración determinada, el entorno reducido de una solución se obtendrá como el entorno usual eliminando las soluciones etiquetadas como tabú (prohibidas durante el periodo tabú). El objetivo principal de etiquetar las soluciones visitadas como tabú es evitar que el proceso de búsqueda alcance un estado en el que se repitan soluciones, es decir, que la búsqueda entre una etapa cíclica. Por ello, se presupone que tras un cierto número de iteraciones la solución se encontrará en una región distinta del espacio objetivo, y puede liberarse del estado tabú a las soluciones antiguas.

Para almacenar estos movimientos prohibidos se hace uso de las denomi-

nadas listas tabu (LT). Mientras que en los orígenes de TS se sugerían LT de longitud (ψ) reducida, actualmente algunas implementaciones ajustan dicho tamaño dinámicamente según una determinada estrategia. Se define como *nivel de aspiración* a aquellas condiciones que, de satisfacerse, permitirán alcanzar una solución aunque esté clasificada como tabú. Una implementación sencilla consiste en permitir alcanzar una solución siempre que mejore a la mejor almacenada, aunque esta esté etiquetada como tabú. De esta forma se introduce cierta flexibilidad en la búsqueda y se mantiene su carácter agresivo.

Algoritmo 1.2.3

Begin TS (ψ)
  $s \leftarrow$ Generar_Solución_Inicial();
  $LT \leftarrow \phi$;
  itera_cont$\leftarrow$0;
  Hacer
    Candidatos$\leftarrow \forall s' \in N(s) : s' \notin LT$;
    $s \leftarrow$ Seleccionar_Mejor_Candidato(Candidatos);
    Actualizar_LT(ψ,itera_cont);
    itera_cont++;
  Mientras (no se cumpla la condición de parada);
  Retornar s;
End TS

Es importante considerar que los métodos basados en búsqueda local requieren de la exploración de un gran número de soluciones en poco tiempo, por lo que reducir al mínimo el esfuerzo computacional de las operaciones que se realizan a menudo se convierte en un aspecto crítico. En ese sentido, la memoria a corto plazo está basada en atributos en lugar de ser explícita, esto es, en lugar de almacenar las soluciones completas (como ocurre en los procedimientos de búsqueda exhaustiva) se almacenan únicamente algunas de sus características. La memoria mediante atributos produce un efecto más útil y beneficioso en la búsqueda, ya que un atributo o grupo de atributos identifica a un conjunto de soluciones. Así, un atributo que fue etiquetado como tabú por pertenecer a una solución visitada anteriormente puede impedir en la iteración actual alcanzar una nueva solución, aunque esta sea muy diferente de la que provocó que el atributo fuese etiquetado como tabú. Esto ocasiona, a largo plazo, que se identifiquen y mantengan

aquellos atributos que inducen una cierta estructura beneficiosa en las soluciones visitadas. La memoria a largo plazo almacena las frecuencias u ocurrencias de atributos en las soluciones visitadas tratando de identificar o diferenciar regiones. Dicha memoria cuenta con dos estrategias asociadas: la intensificación y la diversificación de la búsqueda [GLO97]. La intensificación consiste en regresar a regiones ya exploradas para estudiarlas más a fondo. Para ello se favorece la aparición de aquellos atributos asociados a buenas soluciones encontradas. La diversificación consiste en visitar nuevas áreas no exploradas del espacio de soluciones, para lo cual se modifican las reglas de elección, incorporando a las soluciones atributos que no han sido usados frecuentemente.

El Algoritmo 1.2.3 describe el funcionamiento general de TS. Este tiene como parámetro de entrada la longitud de la lista tabú (ψ), es decir, el tiempo durante el cual un determinado movimiento se considera tabú. Una vez obtenida una solución inicial, se inicializa la lista tabú y el número de iteraciones y comienza el bucle principal. Mientras no se de la condición de parada establecida se realizan los siguientes pasos. Primeramente, se genera un conjunto de soluciones candidatas, vecinas a la actual (s) que no estén incluidas en la lista de movimientos tabú. De dicha lista de movimientos candidatos se genera la mejor solución posible, y sustituye a la solución actual. Posteriormente se actualiza la lista tabú en función de la longitud de memoria previamente establecida, además del contador de iteraciones que se utiliza para el control del bucle.

En algunos problemas se han aplicado con éxito variantes de TS, como los movimientos de influencia [GLO03b] que producen cambios sustanciales en la estructura de las soluciones, o elecciones probabilísticas que realizan la selección de forma aleatoria, etc. Es importante destacar el hecho de que muchas de las técnicas basadas en TS no utilizan los últimos elementos descritos, por lo que son susceptibles de ser mejoradas. Al mismo tiempo, el éxito de TS ha llevado a varios investigadores a desarrollar otras formas de explotar con mayor intensidad sus ideas subyacentes. En este terreno podemos destacar los últimos trabajos de modelos de entrenamiento y aprendizaje tabú [GLO03b].

Procedimientos de Búsqueda basados en Procedimientos Adaptativos Aleatorizados Avaros (GRASP)

Los procedimientos de búsqueda basados en funciones adaptativas aleatorizadas avaras (GRASP) [FEO95, RES03] son meta-heurísticas de propósito general que combinan heurísticas constructivas y de búsqueda local. GRASP consta de

una fase de construcción en la cual se aplica un procedimiento heurístico constructivo para obtener una buena solución inicial, y una fase de mejora mediante un algoritmo de búsqueda local. La mejor de todas las soluciones examinadas se guarda como resultado final.

La fase de construcción se lleva a cabo de una forma iterativa, añadiendo un nuevo elemento en cada paso, tal que en cada iteración la elección del próximo elemento a añadir a la solución parcial viene determinada por una función avara (greedy). Esta función mide el beneficio de añadir cada uno de los elementos según la función objetivo y elegir la mejor. Indicar que esta medida es miope en el sentido que no tiene en cuenta qué ocurrirá en iteraciones sucesivas al realizar una elección, sino únicamente en esta iteración. En cada iteración se actualizan los beneficios obtenidos al añadir el elemento seleccionado a la solución parcial. La estrategia de selección utilizada es pseudo-aleatoria, ya que no selecciona el mejor candidato según la función avara utilizada, sino que, con el objeto de diversificar y no repetir soluciones en dos construcciones diferentes, se construye una lista con los mejores candidatos para posteriormente tomar una al azar. Al igual que ocurre en muchos métodos deterministas, las soluciones generadas por la fase constructiva de GRASP no suelen ser óptimos locales. Dado que dicha selección aleatoria provoca que no se pueda garantizar la optimalidad respecto a la estructura de entorno en la que se está trabajando, se aplica un procedimiento de búsqueda local para mejorar la solución obtenida.

Algoritmo 1.2.4

```
Begin GRASP ()
    Hacer
        s←Construir_Solución_Aleatorizada_Avara();
        Aplicar_Búsqueda_Local(s);
        Memorizar_Mejor_Solución_Encontrada();
    Mientras  (no se cumpla la condición de parada);
    Retornar s;
End GRASP
```

Para evitar invertir demasiado tiempo en la fase de mejora, se suele emplear un procedimiento de intercambio simple. Indicar que GRASP realiza múltiples iteraciones quedándose con la mejor solución, por lo que no resulta especialmen-

te beneficioso detenerse demasiado en mejorar una solución dada. Mediante la realización de un gran número de iteraciones, se consigue un muestreo adecuado del espacio de soluciones. Algunos estudios empíricos han demostrado que la distribución de la muestra generalmente tiene un valor en promedio que es inferior al obtenido por un procedimiento determinista. Sin embargo, la mejor de las soluciones encontradas suele mejorar con una alta probabilidad a la obtenida por el procedimiento determinista.

Las características más relevantes de GRASP son su sencillez, rapidez, y facilidad de implementación [RES03]. Basta con fijar el tamaño de la lista de candidatos y el número de iteraciones para determinar completamente el procedimiento. De esta forma se pueden concentrar los esfuerzos en diseñar estructuras de datos para optimizar la eficiencia del código y proporcionar una gran rapidez al algoritmo, dado que este es uno de los objetivos principales de GRASP. Además, las implementaciones GRASP generalmente son muy robustas, encontrando soluciones de buena calidad en la mayoría de las aplicaciones. Sin embargo, se puede mejorar realizando algunas modificaciones en el procedimiento, como por ejemplo incluir una fase determinista previa a la reconstrucción con el objetivo de ahorrar esfuerzo a la fase siguiente. El Algoritmo 1.2.4 presenta la descripción de alto nivel de GRASP.

Búsqueda en Vecindario Variable (VNS)

La búsqueda en vecindario variable (en inglés Variable Neighborhood Search, VNS) [HAN99] es una estrategia basada en modificar dinámicamente las estructuras de vecindario. Se trata de un procedimiento muy genérico, ya que se permite un alto grado de libertad en cada implementación.

En el paso de inicialización se definen varias estructuras de vecindario, que se pueden escoger arbitrariamente, aunque normalmente se utilizan vecindarios con cardinalidad creciente. Se genera una solución inicial, s, el índice de vecindario (kv), y el algoritmo itera hasta que se alcanza la condición de parada. El bucle principal de VNS está formado por tres fases: agitación, búsqueda local y movimiento. La fase de agitación consiste en tomar una solución s' en el kv-ésimo vecindario de la solución actual s, de forma que s' se convierte en el punto de partida de la búsqueda local, que se puede aplicar a cualquier estructura de vecindario. Al final del proceso de búsqueda local, si la nueva solución obtenida s" es mejor que s la reemplaza y el algoritmo se inicia de nuevo con dicho elemento (s"). En otro caso, kv es incrementado y se inicia una nueva fase de

agitación utilizando un vecindario diferente. El objetivo de la fase de agitación es perturbar la solución de forma que se obtenga una buena solución de partida para la búsqueda local. Dicha solución de partida debería pertenecer a una zona diferente de la solución actual, aunque no demasiado lejana, ya que en este caso podría degenerar en un algoritmo multi-arranque. Además, al elegir *s'* en el vecindario de la mejor solución actual es probable que produzca una solución que mantenga las buenas características de la actual. El proceso de cambiar vecindarios en caso de que no haya mejoras corresponde a una diversificación de la búsqueda. En particular, la elección de vecindarios para incrementar la cardinalidad produce una diversificación progresiva.

La efectividad de esta estrategia de vecindario dinámico puede ser explicado con el hecho de que una mala posición en el espacio de búsqueda para un vecindario, puede ser una buena región para otro vecindario. Por otro lado, una solución que sea óptimo local en un vecindario, probablemente no lo sea para otro. Es decir, las propiedades del espacio suelen variar en otros estructuras de vecindario, por lo que las estrategias de búsqueda pueden comportarse de forma diferente en cada caso.

1.2.2. Técnicas Meta-heurísticas Poblacionales

Algoritmos Evolutivos (EA)

Los algoritmos evolutivos (en inglés Evolutionary Algorithms, EA) [FOG06] se consideran como técnicas de aprendizaje no supervisado que, en general, no requieren de ningún tipo de conocimiento ya que el propio algoritmo es capaz de generar por sí mismo el conocimiento que requiere. Se consideran como técnicas meta-heurísticas de búsqueda que usan operadores probabilísticos y que funcionan como cajas negras, pues no requieren conocimiento específico del dominio para actuar, y que son aplicables a un gran número de aplicaciones. La computación evolutiva [FOG06] interpreta la naturaleza como una inmensa máquina de resolver problemas, y trata de encontrar los principios en que se basa su comportamiento para utilizarlos en los procedimientos de optimización. En la naturaleza todos los seres vivos se enfrentan a problemas que deben resolver con éxito, como obtener alimento o conseguir más luz solar. El origen de esta capacidad está en la evolución producida por la selección natural que favorece la perpetuación de los individuos más adaptados a su entorno. La principal diferencia conceptual entre la selección natural que se produce sin intervención del hombre, y la selección artificial que establecemos en nuestros programas, es que

mientras nosotros podemos seleccionar para la reproducción los seres que más nos interesan, en la naturaleza no existe una inteligencia exterior que determine la dirección de la evolución. Aun así, la evolución se produce.

Algoritmo 1.2.5

Begin EA (P_{tam}, ρ_{cruce}, ρ_{mut})
  P←Inicializar_Población(P_{tam});
  Evaluar(P);
  Hacer
    P'←Aplicar_Cruce(P,ρ_{cruce});
    P''←Aplicar_Mutación(P',ρ_{mut});
    Evaluar(P'');
    P←Seleccionar($P \cup P''$);
  Mientras (no se cumpla la condición de parada);
  Retornar Mejor_Solución(P);
End EA

Los EA han tenido un gran auge en los últimos años, gracias a los avances tećnicos que vinieron tras los primeros estudios de Holland [HOL75]. En muchos de los problemas de optimización, los EA ofrecen buenos resultados en tiempos reducidos mediante la creación y mejora de una población de soluciones (ver Figura 1.2(b)) a las que se aplican los operadores de mutación y cruce. El proceso de evolución se produce mediante operaciones en los cromosomas de ciertos individuos. En el proceso de reproducción tiene lugar la evolución mediante la combinación cromosómica de los progenitores. El proceso por el que se forma el cromosoma del descendiente es conocido como recombinación. También hay que tener en cuenta las mutaciones que pueden alterar dichos códigos. En el proceso de evolución no existe memoria como tal, ya que únicamente se considera la información del periodo anterior, y no de periodos anteriores, lo cual simplifica la forma de implementación. El Algoritmo 1.2.5 describe el patrón general de funcionamiento de un algoritmo evolutivo. Básicamente, este procedimiento trabaja con una población de soluciones (individuos) que una vez inicializados, trabajan de la siguiente forma. Primeramente se aplica un operador de cruce que trata de combinar características de dos o incluso más individuos. Tras ello se aplica el operador de mutación, que consiste en realizar una pequeña variación de determinados individuos para intentar mejorar su calidad. Ambos operadores

se aplican con una determinada probabilidad (ρ_{cruce}, ρ_{mut}). Tras evaluar la población generada tras aplicar los operadores se aplica el proceso de selección, de forma que se obtiene la población que será evaluada en la siguiente generación.

Búsqueda Dispersa (SS)

La búsqueda dispersa (en inglés Scatter Search, SS) [MAR06] es un método evolutivo que ha sido aplicado en la resolución de un gran número de problemas de optimización. Los conceptos y principios fundamentales del método fueron propuestos en la década de los setenta [GLO77]. Recientemente, Martí y col. [MAR06] han descrito las tendencias actuales de esta meta-heurística. SS utiliza estrategias para combinar reglas de decisión, y se basa en el principio de que la información sobre la calidad o el atractivo de un conjunto de reglas, restricciones o soluciones puede ser utilizado mediante la combinación de estas.

SS lleva a cabo el proceso de búsqueda mediante el uso de un conjunto de soluciones (P), denominadas agentes, que, al igual que los algoritmos genéticos, son optimizadas simultáneamente. Sin embargo, mientras que en el proceso de selección en los algoritmos genéticos [GOL89] suelen intervenir un porcentaje muy elevado de la población o toda ella, SS está fundamentada en las elecciones sistemáticas y estratégicas sobre un conjunto relativamente pequeño de soluciones, denominado conjunto de referencia (PR). Este conjunto contiene aquellas soluciones previamente obtenidas que cumplen ciertos criterios de calidad y diversidad. La estructura de SS consta de cinco métodos principales [MAR06].

- un método de diversificación que genera un conjunto de soluciones heterogéneas (P);
- un método de mejora local para transformar una solución candidata en una o más soluciones;
- un método de actualización del conjunto de referencia (PR) que mantiene una porción de las mejores soluciones de P;
- un método de generación de subconjuntos (PR') que opera con el conjunto de referencia (PR), de forma que algunas soluciones son seleccionadas con la idea de combinarlas en una fase posterior;
- un método generador de soluciones que transforma las soluciones construidas por el método de generación de subconjuntos en una o más soluciones combinadas.

El primer aspecto a considerar en SS es la necesidad de desarrollar un genera-

dor de soluciones diversas (población P), cuyo tamaño puede oscilar en torno a P_{tam}=100 soluciones [MAR06], del cual se extrae el llamado *conjunto de referencia* (PR), cuyo tamaño puede oscilar en torno a PR_{tam}=10 soluciones [MAR06]. El criterio para extraer estas soluciones se basa en la calidad y diversidad, de forma que dicho conjunto de referencia suele estar formado a partes iguales entre aquellas soluciones del conjunto diverso con mayor calidad y aquellas, también del conjunto diverso, que se encuentren más lejos (en distancia euclídea) de las soluciones recién incluidas en el conjunto de referencia (PR).

Algoritmo 1.2.6

Begin SS (P_{tam}, PR_{tam})
  P←Generador_Soluciones(P_{tam});
  PR←ϕ;
  Para (cada solución $s \in P$) **hacer**
    s←Aplicar_Mejora(s);
  $PR \leftarrow PR \cup$ (PR_{tam}/2 mejores $\in P$);
  $PR \leftarrow PR \cup$ (PR_{tam}/2 $\in P$ más diversas respecto a PR);
  Hacer
    PR'←Generar_Subconjuntos_Referencia(PR);
    Hacer
      Seleccionar $PR'_i \in PR'$ y etiquetarlo como visitado;
      Combinar soluciones de PR'_i;
      Para (cada solución $s' \in PR'_i$) **hacer**
        s''←Aplicar_Mejora(s');
        Si (s'' mejora a la peor de PR) **hacer**
          Sustituir dicha solución de PR por s'';
          Eliminar PR'_i;
    Mientras (quede algún $PR'_i \in PR'$ sin evaluar);
  Mientras (no se cumpla la condición de parada);
  Retornar Mejor_Solución(P);
End SS

Otro aspecto importante en SS es la recombinación de soluciones que permiten actualizar el conjunto de referencia, mediante la inclusión de soluciones mejores. Esta actualización se puede realizar en función de la calidad de las nue-

vas soluciones, aunque también es posible hacerlo considerando la diversidad. La recombinación de soluciones se lleva a cabo entre los individuos del conjunto de referencia, de forma que dicho conjunto se divide en subconjuntos de dos o más agentes. Estos subconjuntos se combinan de forma que las soluciones obtenidas se introducen directamente en el conjunto de referencia, o bien se almacenan temporalmente en una lista hasta que se lleven a cabo todas las combinaciones, para posteriormente analizar cuáles son las que se introducen. El último aspecto a considerar es la mejora de soluciones que se consigue mediante algún procedimiento de búsqueda local que trate de mejorar la calidad de las soluciones del conjunto de referencia, así como de las combinadas antes de la inclusión. El Algoritmo 1.2.6 detalla el funcionamiento de SS.

Un factor a tener en cuenta a la hora de implementar SS es que la condición de parada no se controla de forma previa como en la mayoría de los algoritmos, sino que SS se detiene cuando no existen nuevos elementos a combinar dentro del conjunto de referencia. Para resolver este inconveniente se puede realizar un bucle que permita reconstruir el conjunto de referencia cuando este ya haya sido utilizado. Así, si el número de evaluaciones previamente establecido no se ha excedido, se puede regenerar dicho conjunto dejando la mitad superior ($PR_{tam}/2$ mejores) y eliminando la mitad inferior ($PR_{tam}/2$ peores). Después, se genera un conjunto P como al comienzo del algoritmo, del que se extraen únicamente las $PR_{tam}/2$ soluciones más diversas con respecto a las ya existentes. De esta forma se puede continuar el proceso de búsqueda mediante un nuevo conjunto de referencia en el que mantenemos las soluciones de calidad y renovamos las debidas a diversidad.

Algoritmos Meméticos (MA)

Dentro de las técnicas heurísticas más recientes encontramos los algoritmos meméticos (en inglés Memetic Algorithms, MA) [MOS00]. Los MAs presentan aspectos de especial similitud con los algoritmos evolutivos, sobre todo en lo referente a las mejoras individuales de las soluciones junto con procesos cooperativos y competitivos de tipo poblacional. Los componentes de las poblaciones en MAs no se denominan individuos, sino agentes. En cada generación se actualiza la población de agentes, usando para tal fin una población temporal obtenida mediante la recombinación de algunos agentes seleccionados. La selección se encarga de elegir una muestra de los mejores agentes contenidos en la población actual haciendo uso de una determinada función objetivo. Por otro lado, el pro-

ceso de reemplazo incide en el aspecto competitivo, limitando el tamaño de la población y permitiendo de esta forma que otros nuevos agentes puedan entrar y diversificar la búsqueda. Tanto la selección como el reemplazo son procesos competitivos en los que únicamente varía la distribución de agentes existentes, pero no se crean nuevos, ya que esto corresponde a la fase de reproducción. Los operadores utilizados son los de recombinación y mutación. La recombinación es responsable de llevar a cabo los procesos de cooperación entre dos o más agentes haciendo uso de la información del grupo de agentes recombinados. El operador de mutación consiste en modificar las características de un determinado agente. El empleo de estos meta-operadores es uno de los rasgos más distintivos de los MAs. Concretamente, dichos meta-operadores iteran la aplicación del operador de mutación, conservando los cambios que llevan a una mejora en la calidad del agente, motivo por el cual son denominados optimizadores locales. Estos optimizadores locales pueden emplearse en diferentes fases de la reproducción. Existen diferentes alternativas de diseño. Así, por ejemplo, pueden usarse tras la aplicación de otros operadores de recombinación y mutación, aplicarse sólo a un subconjunto de los agentes, o ser únicamente aplicado al final del ciclo reproductivo. En cualquier caso, el empleo de optimización local es un elemento diferenciador de los MA con respecto a los EA pero no el único [MOS00].

Optimización mediante Colonias de Hormigas (ACO)

La optimización mediante colonias de hormigas (en inglés Ant Colony Optimization, ACO) [DOR99] se fundamenta en el comportamiento real de las hormigas. Este comportamiento se puede resumir en que las hormigas son capaces de encontrar la ruta más corta entre el lugar donde se encuentra el alimento y su escondite subterráneo [DEN90]. Cuando las hormigas recorren el camino en busca de comida, depositan en el suelo una sustancia química inodora, conocida como feromona. A la hora de seleccionar el lugar hacia donde ir, eligen con mayor probabilidad aquellas rutas cuya concentración de feromona es mayor. Este proceso cooperativo permite que la búsqueda se dirija hacia las rutas más cortas. Las soluciones se van construyendo de forma incremental mediante la agregación de componentes a soluciones parciales. Cuando se trata un determinado problema de optimización combinatoria, las restricciones del problema son construidas mediante un procedimiento constructivo de hormigas, de forma que en cada paso el proceso constructivo solamente puede añadir componentes factibles a la solución parcial actual. En la mayoría de las aplicaciones, las colonias

han sido implementadas para construir soluciones factibles, aunque bajo determinadas circunstancias se permite la construcción de soluciones no factibles. En [DOR99] se puede encontrar más información sobre el funcionamiento de dicha técnica.

1.3. Conclusiones

En este capítulo se han comentado aspectos básicos relacionados con los problemas de optimización combinatoria. Debido a la gran complejidad de las aplicaciones reales en las que aparecen los problemas combinatorios, las técnicas deterministas no permiten obtener soluciones factibles en tiempos de ejecución aceptables. Bajo dichas circunstancias, una buena alternativa consiste en aplicar técnicas heurísticas, que consisten en procedimientos sistemáticos de prueba que ofrecen soluciones óptimas o aproximadas para problemas donde el espacio de soluciones es indeterminado o lo suficientemente amplio como para que no pueda ser evaluado exhaustivamente en un tiempo aceptable. Dado que el objetivo principal de este trabajo se encuentra en el diseño de heurísticas híbridas, en este capítulo se han descrito brevemente las principales meta-heurísticas encontradas en la literatura, incluidas aquellas que son utilizadas en los procedimientos propuestos que se describen en capítulos posteriores.

En el Capítulo 2 se extienden los conceptos generales de optimización presentados en el presente capítulo, con el objetivo de resolver problemas de optimización multi-objetivo haciendo uso del concepto de optimización basada en frentes de Pareto.

Capítulo 2

Optimización Multi-objetivo: Conceptos y Técnicas

Como describimos en el Capítulo 1, la optimización consiste en encontrar la solución óptima de entre un conjunto finito de soluciones alternativas. Mientras que en el contexto mono-objetivo la calidad de una determinada solución viene dada directamente por el valor de aptitud en la función objetivo. En el contexto multi-objetivo no existe un solo óptimo global, sino varios que optimizan todos los objetivos del problema. En el presente capítulo se describen los conceptos necesarios parar tratar problemas multi-objetivo haciendo uso de optimización basada en frentes de Pareto. Además, se ofrece una breve descripción de las principales técnicas heurísticas multi-objetivo encontradas en la literatura, clasificandolas en función de si están basadas en computación evolutiva, en búsqueda local, o combinan aspectos de diferentes técnicas (técnicas híbridas).

2.1. Introducción a la Optimización Multi-objetivo Basada en Frentes de Pareto

La mayoría de problemas de optimización reales suelen tener varios óptimos (máximos o mínimos), siendo uno de ellos el óptimo global, mientras que el resto son óptimos locales en el sentido de que es posible definir una vecindad alrededor de ellos en la cual son óptimos globales [HOS95]. Sin embargo, en muchos casos todos los óptimos absolutos tienen la misma importancia, bien por tener el mismo valor numérico, o bien porque no se pueda establecer un criterio que permita decidir cuál de ellos es mejor. Esto sucede en los denominados problemas de optimización multi-objetivo (MOPs) [COE98]. Por ejemplo, imaginemos un problema real como el de un distribuidor de mercancias que trata de minimizar el tiempo necesario para su distribución y, simultáneamente, el combustible gastado. No se puede cambiar más tiempo por menos combustible, ni viceversa, sino que los dos se consideran de la misma importancia. Habrá soluciones (90 minutos y 13 litros) mejores que otras (100 minutos y 14 litros), pero en algunos casos, no se podrá comparar (90 minutos y 13 litros ó 100 minutos y 12 litros).

Los problemas en los que se obtienen múltiples soluciones de forma que ninguna de ellas se pueda probar mejor que otras suelen tratarse mediante técnicas multi-objetivo específicas, destacando sobremanera las conocidas como técnicas de optimización basadas en frentes de Pareto [GOL89, COE98]. Dicho concepto, que se describe formalmente con posterioridad, trata de encontrar todas las soluciones pertenecientes al conocido como frente Pareto-óptimo [GOL89]. Dicho frente se puede definir como el conjunto de soluciones que no pueden ser mejoradas en todos los objetivos o igualadas en unos y mejorada en otros por ninguna otra solución que también cumpla las restricciones del problema.

Cabe indicar que la utilización de técnicas heurísticas para resolver este tipo de problemas resulta incluso más conveniente al incrementarse la complejidad con respecto a las formulaciones mono-objetivo. Así pues, dichas meta-heurísticas multi-objetivo (MOMHs) buscan encontrar el conjunto de soluciones Pareto-óptimas o un conjunto representativo de este, de forma que el usuario pueda realizar consideraciones de más alto nivel para elegir una o varias de ellas, tal y como se describe en la Figura 2.1.

A modo de ejemplo, la Figura 2.2 se puede observar como el conjunto de soluciones (P) obtenidas por un determinado algoritmo se aproximan bastante al frente óptimo de soluciones de los problemas bi-objetivo ZDT1 y ZDT3 [ZIT99].

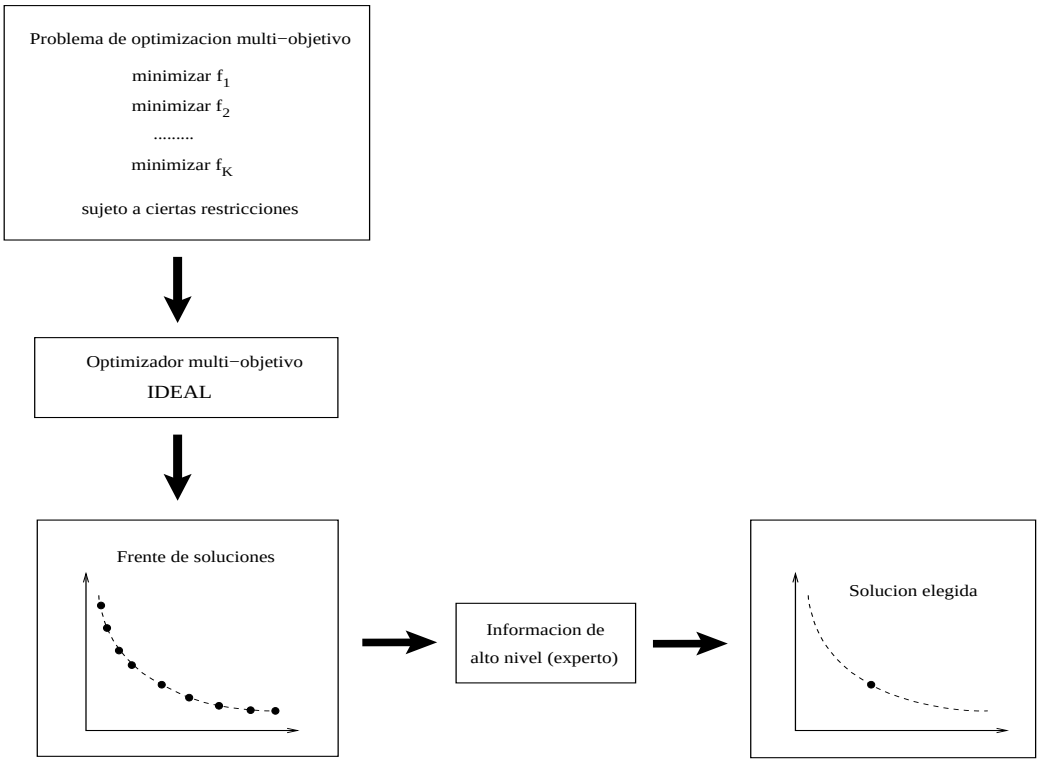

Figura 2.1: Esquema de un procedimiento de optimización multi-objetivo ideal.

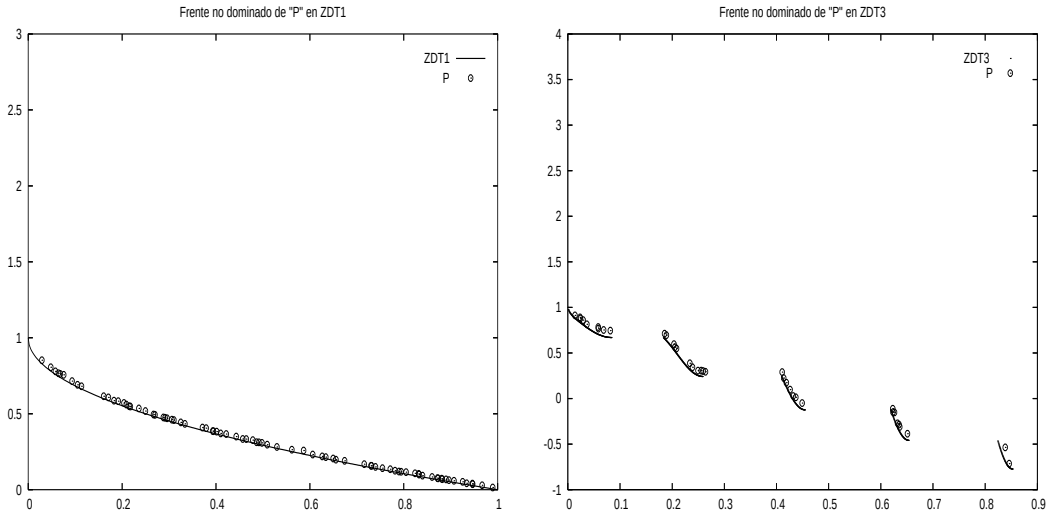

Figura 2.2: Aproximaciones a los frentes Pareto-óptimos (a) ZDT1, (b) ZDT3.

Definición 2.1. La optimización multi-objetivo (MOO) es el proceso de búsqueda de una o más soluciones que satisfagan de forma simultánea todas las restricciones, y optimicen una determinada función que relaciona el espacio de decisión al que pertenecen las soluciones con el espacio objetivo. Para un problema con $K \geq 2$ objetivos, el proceso de optimización consiste en minimizar o maximizar la función objetivo f:

$$\text{minimizar/maximizar}(f_k(s)), \quad \forall k \in [1, K]$$

Cada vector de decisión o solución s representa las cualidades numéricas para un problema de optimización multi-objetivo. A diferencia del caso mono-objetivo, en el contexto multi-objetivo la función objetivo (f) transforma los vectores de decisión desde el espacio de búsqueda hacia un espacio objetivo de K dimensiones $Z \in \Re^K$, $z = f(s)$, $f(s) = \{f_1(s), f_2(s), ..., f_K(s)\}$, $z \in Z$, $s \in F$.

Dado un MOP con $K \geq 2$ objetivos, en lugar de retornar un valor escalar a cada solución, se establecen relaciones parciales de acuerdo a las relaciones de Pareto-dominancia, tal y como se detalla a continuación.

Definición 2.2. Relaciones de Pareto-dominancia. Sean s y s' dos vectores de decisión (soluciones). Las relaciones de dominancia entre vectores de decisión para un problema de minimización son:

s *domina a* s' $(s \prec s')$ *sii* $\forall k \in [1,K]\ \exists k' \in [1,K] : f_k(s) \leq f_k(s') \wedge f_{k'}(s) \neq f_{k'}(s')$

s, s' son *indiferentes o incomparables* $(s \sim s')$ *sii* $s \not\prec s' \wedge s' \not\prec s$

Definición 2.3. Solución Pareto-óptima. Una solución s se denomina *Pareto-óptima sii* $\nexists s' \subseteq F$, tal que $s' \prec s$. Todas las soluciones Pareto-óptimas definen el *conjunto Pareto-óptimo*.

Definición 2.4. Solución no dominada. Una solución $s \subseteq$ F es *no dominada* con respecto al conjunto S'$\subseteq$F *sii* $\nexists$s'$\subseteq$S', tal que $s' \prec s$.

Definición 2.5. Conjunto no dominado. Dado un conjunto de soluciones S, tal que $S \subseteq F$, la función $ND(S)$ devuelve el conjunto de soluciones no-dominadas de S: $ND(S) = \{\forall s \subseteq S\ \nexists s' \subseteq S : s' \prec s\}$

Las soluciones Pareto-óptimas no pueden ser mejoradas en ningún objetivo

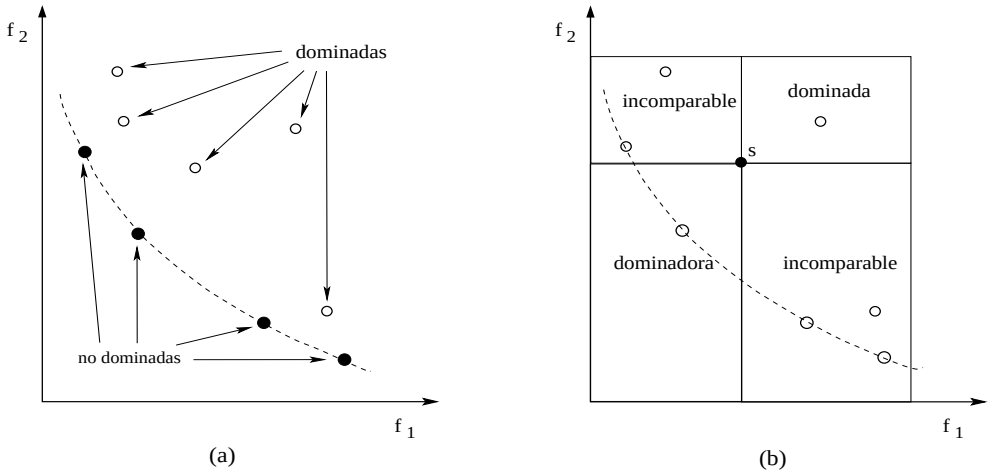

Figura 2.3: Relaciones de Pareto-dominancia.

sin causar degradación en al menos otro objetivo, por lo cual se consideran como óptimos globales. No obstante, análogamente a los problemas de optimización de un solo objetivo, existen óptimos locales que son conjuntos no dominados en ciertas áreas locales. La Figura 2.3 muestra un ejemplo de dominancia entre soluciones. En la Figura 2.3(a) se puede observar en puntos rellenos un conjunto de soluciones no dominadas. En la Figura 2.3(b) se muestra la relación entre soluciones. Así, la solución de referencia s, puede observarse como hay soluciones que dominan a s, otras que son dominadas por s, y otras que son incomparables.

A la hora de implementar un algoritmo multi-objetivo se deben considerar los siguientes objetivos de diseño:

- se debe minimizar la distancia entre las soluciones no dominadas encontradas por el algoritmo y la frontera de Pareto;
- las soluciones encontradas deben de representar la totalidad de la región Pareto-óptima;
- la distribución de las soluciones encontradas a lo largo de la región Pareto-óptima debería ser lo más uniforme posible.

Por todo ello, a la hora de evaluar el comportamiento de cualquier algoritmo, resulta necesario hacer uso de medidas adecuadas para evaluar cada uno de los aspectos anteriormente descritos, de forma que las conclusiones obtenidas sean robustas. En la siguiente sección, analizaremos algunas de las métricas de

rendimiento utilizadas en este contexto.

En la formulación mono-objetivo la métrica que determina la calidad de las soluciones viene dada directamente por el objetivo a optimizar. Sin embargo, en el caso multi-objetivo es necesario establecer criterios específicos para determinar la calidad de los frentes de soluciones obtenidos considerando los objetivos a optimizar. Diferentes autores [DEB01, VEL03] han analizado diversas métricas de rendimiento para problemas multi-objetivo, estableciendo dos grandes categorías de métricas.

En la primera categoría se incluyen aquellas métricas que determinan la proximidad de las soluciones al frente Pareto-óptimo. Dentro de esta categoría se incluyen la métrica del ratio de error, cobertura de frentes, distancia generacional, etc. Todas esas métricas inicialmente propuestas para problemas bi-objetivo pueden adaptarse fácilmente a problemas con más objetivos.

En la segunda categoría, se encuentran aquellas métricas que permiten establecer la diversidad del frente de soluciones obtenido. El inconveniente de estas métricas es que las medidas de dispersión no pueden aplicarse directamente a problemas con tres o más objetivos, debido a que un conjunto de soluciones con una buena métrica de espaciado no significa necesariamente una buena distribución de soluciones en todo el frente Pareto-óptimo. Otra alternativa es la métrica de la dispersión máxima, que mide la distancia euclidea entre soluciones extremas, aunque no revela la distribución de las soluciones intermedias.

Zitzler y col. [ZIT99] aplicaron de forma conjunta la métrica de cobertura de conjuntos junto con la métrica del área del espacio cubierto (también llamada hiper-volumen o medida de Lebesgue) en el problema de la mochila con 2, 3 y 4 objetivos. A partir de dicho trabajo han sido muchos los autores que han considerado el uso conjunto de estas métricas para evaluar la calidad de los frentes obtenidos en problemas multi-objetivo. En esta tesis utilizamos estas dos métricas a la hora de evaluar el rendimiento de las diferentes implementaciones multi-objetivo para los problemas de prueba utilizados. Debido a que los objetivos a optimizar en los problemas de repartición de grafos y de optimización de redes de distribución de agua son diferentes, ofreceremos una descripción detallada por separado para cada problema.

2.2. Técnicas Heurísticas de Optimización Multi-objetivo

Actualmente, en la literatura se pueden encontrar aproximadamente medio centenar de meta-heurísticas multi-objetivo (MOMH) [EMO]. Sin embargo, muchas de ellas están limitadas a frentes de Pareto con ciertas características (p.ej., convexos) y suelen requerir un punto inicial de búsqueda. Dichas técnicas de optimización se pueden agrupar en tres grandes categorías: algoritmos evolutivos multi-objetivo (en inglés Multi-objective Evolutionary Algorithms, MOEAs), algoritmos multi-objetivo basados en búsqueda local (en inglés Multi-Objective Local Search Algorithms, MOLSAs), y meta-heurísticas híbridas multi-objetivo (en inglés hybrid MOMHs, hMOMHs).

2.2.1. Algoritmos Evolutivos Multi-objetivo

Dentro de las técnicas meta-heurísticas mejor situadas para explorar el frente Pareto-óptimo en MOPs se encuentran los algoritmos evolutivos (EAs). Debido a su inherente paralelismo y a su capacidad para explotar las similitudes de las soluciones mediante recombinación, los EAs son capaces de aproximar el frente de Pareto en una sola ejecución. El potencial de los algoritmos evolutivos para resolver MOPs se remonta a la tesis doctoral de Rosenberg [RSB67], en la cual se sugería el uso de algoritmos genéticos en este dominio. Sin embargo, el primer intento real por extender un algoritmo evolutivo a problemas multi-objetivo es el Vector Evaluated Genetic Algorithm (VEGA) [SCF85].

Históricamente podemos considerar que han existido dos generaciones de algoritmos evolutivos multi-objetivo:

- Primera generación: formada por algoritmos relativamente simples que hacian uso de enfoques rudimentarios, como funciones agregativas lineales [HAJ92], así como otras más complejas, como funciones de jerarquización de Pareto y nichos [MAH95].
- Segunda generación: se introduce el concepto de elitismo, que consiste en un mecanismo empleado para asegurar que los individuos con mejores valores de aptitud pasarán directamente a la siguiente generación, evitando de esta forma el ruido provocado por los operadores genéticos. Las dos formas principales de implementar el elitismo son la selección (μ+λ), y el uso de poblaciones secundarias [LAU01]. Los algoritmos de segunda generación enfatizan la eficiencia computacional.

En la última década se ha puesto de manifiesto experimentalmente el efecto beneficioso de introducir el elitismo en el diseño de los MOEAs [ZIT00]. Sin embargo, los EAs, sobre todo si se utilizan procedimientos elitistas de selección en cada generación, tienden a converger a una única solución óptima del espacio de búsqueda. Por lo tanto, para resolver problemas de optimización multi-modal [KUM03] hay que introducir alguna estrategia para mantener la diversidad de soluciones en la población. Se han propuesto diferentes estrategias para conseguir este propósito, destacando entre otros los métodos de nichos (niching methods), métodos de repartición de aptitud (fitness sharing), métodos de densidad (crowding), métodos de borrado (clearing), etc. [SAR98].

Estrategias Basadas en Funciones de Agregación

Una de las primeras estrategias que se propusieron para lidiar con varios objetivos consiste en combinarlos mediante una función de suma de pesos [HAJ92, COE98]. Al proceso de combinar objetivos en una sola función se le denomina normalmente función de agregación y se ha utilizado en diversas ocasiones con relativo éxito en problemas en los cuales el comportamiento de las funciones objetivo se conoce adecuadamente. Este método consiste en sumar todas las funciones objetivo usando diferentes pesos para cada una de ellas. Esto significa que nuestro problema con objetivos múltiples se transforma en un problema de optimización simple de la forma:

$$min \sum_{k=1}^{K} (\sigma_k \cdot f_k(s)) \quad : \quad \sum_{k=1}^{K} \sigma_k = 1 \tag{2.1}$$

donde σ_k representa la importancia relativa del k-ésimo objetivo.

Esta técnica es muy eficiente desde el punto de vista de recursos de cómputo, y puede usarse para generar una solución que sirva como un punto inicial para otras técnicas. Puesto que los resultados obtenidos usando esta función pueden variar significativamente en función de los pesos, y puesto que se sabe normalmente muy poco acerca de la forma más adecuada de seleccionar dichos coeficientes, se sugiere resolver el mismo problema usando diferentes valores de σ_k. Debe advertirse que los pesos no reflejan proporcionalmente la importancia relativa de los objetivos, sino que son sólo factores que, al ser modificados, permiten localizar puntos diferentes en el frente de Pareto, por lo que el diseñador todavía tiene que afrontar la decisión de tener que elegir la solución más apropiada en base a su intuición. Para que los métodos numéricos puedan usarse

para buscar el mínimo de la expresión, esta localización de puntos depende no sólo de σ_k, sino también de las unidades en las que se expresen las funciones. El principal problema es cómo determinar los pesos apropiados cuando no tenemos suficiente información acerca del problema. En este caso, cualquier punto óptimo obtenido sería una función de los coeficientes usados para combinar los objetivos. La mayor parte de los investigadores usan una simple combinación lineal de objetivos y posteriormente generan la superficie de compromisos mediante la variación de los pesos [CZY98].

Vector Evaluated Genetic Algorithm (VEGA)

Con el objetivo de superar las dificultades de los métodos agregados, Schaffer propuso en 1985 el Vector Evaluated Genetic Algorithm (VEGA) [SCF85]. En la literatura VEGA es considerada como una de las técnicas más populares dentro de las que no incorporan el concepto de óptimo de Pareto. VEGA fue una extensión del programa GENESIS [GRE84] que permitía el manejo de objetivos múltiples mediante la extensión del GA simple realizando un proceso de selección diferente. En concreto, en cada iteración se genera un cierto número de sub-poblaciones efectuando una selección proporcional de acuerdo a la función objetivo en turno. De tal forma, para un problema con K objetivos se generan K sub-poblaciones de tamaño P_{tam}/K cada una (suponiendo que P_{tam} es el tamaño total de la población). Estas sub-poblaciones se mezclarán entre sí a fin de obtener una nueva población del mismo tamaño, en la que el GA podrá aplicar los operadores de cruce y mutación de la forma usual. Schaffer se percató de que las soluciones generadas por su sistema eran no dominadas localmente, ya que la no dominancia se limitaba a la población en proceso. Un individuo que es no dominado en una cierta generación puede resultar dominado por otro individuo que emerja en una generación posterior. También advirtió un problema que en genética se conoce como especiación, y que consiste en que ciertas especies con características específicas evolucionan en direcciones distintas. Este problema se origina porque esta técnica selecciona individuos que son excelentes en una cierta dimensión, pero no necesariamente en otras. El peligro potencial detectado por Schaffer es que podremos tener buenas soluciones compromiso, pero que no sobrevivirán bajo este esquema de selección, puesto que no son las mejores en ninguno de los objetivos del problema, sino sólo moderadamente buenas en todos. La especiación es un fenómeno indeseable en este contexto ya que hace más difícil encontrar soluciones compromiso. Schaffer sugirió algunas heurísticas

para abordar este problema. Por ejemplo, propuso usar una selección mediante preferencias heurísticas para los individuos no dominados en cada generación a fin de proteger a nuestros cromosomas mediatizados. Así mismo, propuso motivar el cruce entre las diferentes especies agregando alguna heurística específica en vez de hacer cruce aleatorio como en el GA tradicional.

Con posterioridad al trabajo presentado por Schaffer, Richardson [RIC89] realizó un estudio en el cual advirtió que el hecho de barajar y mezclar todas las sub-poblaciones era equivalente a promediar las aptitudes de cada uno de los objetivos, por lo que la aptitud esperada resultante corresponde a una combinación lineal de los objetivos, donde los pesos dependen de la distribución de la población en cada generación. La principal consecuencia de esto se da cuando la superficie de compromiso es cóncava, ya que una simple combinación lineal de los objetivos no podrá encontrar algunos de los puntos de esta superficie, independientemente del conjunto de pesos que se use para combinarlos [RIC89].

Multi-objective Genetic Algorithm (MOGA)

Tras estos primeros métodos en optimización multi-objetivo, Goldberg propuso el primer algoritmo que usa repartición de aptitud basada en el óptimo de Pareto [GOL89], cuyo objetivo era el de resolver los problemas de la técnica de Schaffer. Goldberg sugirió el uso de asignación de jerarquías y selección basada en no dominancia para trasladar la población hacia el frente Pareto-óptimo. La idea básica es encontrar el conjunto de soluciones en la población que sean no dominadas (en el sentido de Pareto). A estas soluciones se les asigna entonces la jerarquía más alta y se eliminan de la población. De la población restante se obtiene otro conjunto de soluciones no dominadas, a las que se les asigna la siguiente jerarquía en turno. Este proceso se repite hasta que toda la población está jerarquizada.

Fonseca y Fleming [FON93] propusieron el llamado algoritmo genético multi-objetivo (en inglés Multi-objective Genetic Algorithm, MOGA). En esta técnica la jerarquía de un cierto individuo se determina en función del número de cromosomas por los cuales es dominado en la población actual. Así pues, dado un individuo s en la generación actual, el cual es dominado por r individuos, su posición actual en la jerarquía puede obtenerse mediante:

$$jerarquía(s) = 1 + r \tag{2.2}$$

A todos los individuos no dominados se les asigna la jerarquía 1, mientras

que a los dominados se les penaliza de acuerdo a la densidad de la población en la región correspondiente de la superficie de compromiso. La asignación de aptitud se efectúa de la manera siguiente:

- se ordena la población de acuerdo a la jerarquía;
- se asigna la aptitud a los individuos interpolando entre el mejor (jerarquía 1) y el peor (jerarquía $JQ \leq P_{tam}$) en la forma propuesta por Goldberg [GOL89], de acuerdo a alguna función que suele ser lineal, aunque no necesariamente;
- se promedian las aptitudes de los individuos de la misma jerarquía, de manera que todos ellos sean muestreados con la misma probabilidad.

Goldberg y Deb [GOL91] demostraron que este tipo de asignación de aptitud suele producir una gran presión de selección que puede llevar a una convergencia prematura. Para evitar este problema, Fonseca y Fleming [FON93] usaron un método de formación de nichos para distribuir la población sobre el frente de Pareto pero en vez de efectuar distribución de aptitud en función de los valores de los parámetros, se propuso que esta estuviera en función de los valores de las funciones objetivo. En MOGA la repartición de aptitud se realiza en el espacio de los valores de las funciones objetivo, lo que significa que dos vectores diferentes con los mismos valores para sus funciones objetivo no pueden existir simultáneamente en la población usando esta técnica. Esto es aparentemente indeseable porque este es precisamente el tipo de soluciones que normalmente estamos buscando, aunque debe decirse que en la práctica la técnica de Fonseca y Fleming funciona aceptablemente [COE98]. MOGA es una buena técnica, eficiente y relativamente fácil de implementar aunque, al igual que todas las demás técnicas de jerarquización de Pareto, su rendimiento depende en gran medida de una selección adecuada del factor de repartición de aptitud.

Non-dominated Sorting in Genetic Algorithm (NSGA)

Con posterioridad a MOGA, Srinivas y Deb [SRI94] propusieron el algoritmo genético de ordenación no dominada (en inglés Non-dominated Sorting in Genetic Algorithm, NSGA) . Su funcionamiento se basa en el uso de varias capas de clasificación de los individuos. Antes de efectuar la selección, la población es jerarquizada en base a la no dominación, de forma que todos los individuos no dominados se clasifican en una misma categoría con un valor arbitrario de aptitud. Para mantener la diversidad de la población se lleva a cabo una repartición de aptitud entre estos individuos clasificados usando los valores arbitrarios de

aptitud previamente definidos. Posteriormente, este grupo de individuos clasificados se ignora, y se evalúa otra capa de individuos no dominados. Este proceso se repite hasta que todos los individuos de la población están clasificados. Esta técnica utiliza selección proporcional, en concreto la variante denominada Stochastic Remainder [BOO82]. Puesto que los individuos en el primer frente tienen el máximo valor de aptitud, siempre obtienen más copias que el resto de la población, lo que conlleva a una rápida convergencia de la población hacia las regiones no dominadas. La repartición de aptitud ayuda a distribuir los individuos sobre esta región. La eficiencia de NSGA yace en la forma en que los objetivos múltiples de un problema se reducen a una función de aptitud arbitraria usando un procedimiento de ordenamiento basado en no dominancia. La asignación de aptitud se lleva a cabo en base a los valores de los parámetros en lugar de considerar los valores de las funciones objetivo, con lo cual se persigue asegurar que se lleve a cabo una mejor distribución de los individuos, así como permitir que existan múltiples soluciones equivalentes. Sin embargo, esta técnica es más ineficiente que MOGA, tanto computacionalmente como en términos de la calidad de los frentes de Pareto que produce, y además es más sensible al factor de repartición de aptitud [COE98].

Niched Pareto Genetic Algorithm (NPGA)

Casi simultáneamente a NSGA, Horn y col. [HOR94] propusieron un algoritmo genético basado en nichos y optimización de Pareto (en inglés Niched Pareto Genetic Algorithm, NPGA). NPGA lleva a cabo un proceso de selección mediante torneo basado en dominancia de Pareto. En vez de limitar la comparación a dos individuos, se usa un conjunto más grande (típicamente de 10 individuos) para efectuar un torneo en el cual el individuo no dominado resultará triunfador. Si existe un empate entre individuos (no se dominan entre sí) entonces se utiliza asignación de aptitud [SAR98]. Normalmente se deben usar poblaciones considerablemente mayores de lo usual para hacer tolerable el ruido producido por la técnica de selección utilizada para los nichos que emergen de la población. Horn y col. utilizaron la asignación de aptitud en el dominio de las funciones objetivo y sugirieron el uso de una métrica en la que se combinan los dominios de las variables de decisión con el de las funciones objetivo lo que condujo a lo que ellos denominaron asignación de aptitud anidada. Puesto que esta técnica no aplica la selección de Pareto a toda la población, sino sólo a un segmento de ella en cada iteración, su ejecución resulta muy rápida y produce buenos fren-

tes de Pareto que suelen mantenerse durante un gran número de generaciones. El inconveniente de esta técnica es que se necesita ajustar cuidadosamente el tamaño del torneo, ya que de él depende en gran medida su desempeño.

Strength Pareto Evolutionary Algorithm (SPEA)

Propuesto por Zitzler y Thiele, el algoritmo evolutivo de fuerza de Pareto (en inglés Strength Pareto Evolutionary Algorithm, SPEA) [ZIT99] se incluye dentro de los MOEAs de segunda generación. Como comentamos anteriormente, los MOEAs de segunda generación destacan por el uso del elitismo, bien mediante una estrategia de selección (μ+λ) o por el uso una población secundaria [LAU01]. Al igual que otras técnicas de optimización multi-objetivo hasta la época, SPEA utiliza el concepto de dominancia de Pareto para asignar la función de aptitud a los individuos, almacena las soluciones no dominadas externamente (elitismo), y utiliza un método de clustering para reducir el número de soluciones no dominadas almacenadas en el conjunto externo sin destruir las características del frente. En particular, SPEA resultaba innovador por los siguientes aspectos:

- combina los aspectos anteriores en un único algoritmo;
- la aptitud de un individuo viene determinada únicamente por las soluciones almacenadas en el conjunto externo siendo irrelevante el hecho de que un miembro de la población domine a otro;
- todas las soluciones del conjunto externo participan en el proceso de selección;
- incluye un nuevo método de nichos para preservar la diversidad de la población, el cual no requiere ningún parámetro de distancia.

SPEA resultó ser un algoritmo muy influyente ya que, además de incluir los aspectos detallados anteriormente, mejoraba el rendimiento de los utilizados hasta la época, en gran medida gracias a que era el único método que hacía uso explícito del concepto de elitismo (mediante el uso del archivo externo). Sin embargo también presentaba sus principales debilidades, entre las que destacaba el hecho de que a la hora de realizar la asignación de aptitud, aquellos individuos que eran dominados por los mismos miembros del archivo tenían el mismo valor de aptitud. Esto significa que en el caso en el que el archivo contenga un solo individuo, todos los miembros de la población tienen el mismo ranking, independientemente de si se dominan entre ellos o no. Como consecuencia, la presión

selectiva se ve rebajada sustancialmente, y en este caso particular, SPEA puede llegar a comportarse como un algoritmo de búsqueda aleatorio.

Strength Pareto Evolutionary Algorithm 2 (SPEA2)

Poco tiempo después de la presentación de SPEA, y con el objetivo de superar las limitaciones que algoritmos posteriores pusieron de manifiesto, Zitzler y col. presentaron SPEA2 [ZIT01]. Las principales aportaciones de SPEA2 respecto a SPEA se pueden resumir a continuación:

- utiliza un sistema de asignación de aptitud mejorado que tiene en cuenta, para cada individuo, el número de soluciones que este domina y el número por las que es dominado;
- presenta una nueva técnica de estimación de densidad por el vecino más cercano que permite una guía más precisa por el espacio de búsqueda;
- hace uso de un nuevo sistema de truncamiento del archivo que garantiza la preservación de las soluciones extremas.

Non-dominated Sorting in Genetic Algorithm II (NSGA-II)

Al igual que ocurrió con SPEA, el método NSGA descrito anteriormente se vió superado por los nuevos procedimientos. Pese a que en su momento NSGA obtuvo gran reconocimiento y fue objeto de multitud de comparativas y referencias, este método tenía varias debilidades, entre las que destacaban:

- Alto coste computacional del procedimiento de clasificación de las soluciones no dominadas: el coste del algoritmo de clasificación de NSGA era muy elevado en términos computacionales para tamaños de población grandes.
- Falta de elitismo: los resultados obtenidos denotaron que el elitismo puede acelerar el rendimiento de los GA significativamente, además de evitar la pérdida de buenas soluciones por efectos aleatorios;
- Necesidad de especificar un parámetro de sharing (*share*) [SAR98]. Los mecanismos tradicionales para mantener la diversidad de la población se habían centrado en el mecanismo de repartición de aptitud. El principal problema del mismo, como se indicó en su momento, radica en la necesidad de especificar el parámetro *share*.

Por esta razón Deb y col. propusieron NSGA-II [DEB00], el cual ha conseguido resultados muy buenos en numerosos estudios con diferentes problemas de test y métricas de rendimiento. En NSGA-II se propone un nuevo método de comparación *crowded* que resuelve los puntos débiles de NSGA. Se puede decir que NSGA-II utiliza repartición de aptitud. Ahora bien, cuando al añadir frentes rebasamos el tamaño máximo permitido, se eliminan las soluciones peor diversificadas del último frente. Entonces, desde este punto de vista estamos realizando un procedimiento de aclarado, el cual está supeditado a la relación de dominancia como se extrae de la definición de nuestro operador de comparación. En conclusión, al igual que SPEA presentaba un funcionamiento híbrido entre *fitness-sharing* y *crowding*, NSGA-II utiliza como mecanismo de niching una combinación de *fitness-sharing* y *clearing* [MAH95, SAR98].

Pareto-envelope Based Selection Algorithm (PESA)

Corne y col. [COR00] propusieron un algoritmo de selección basado en Pareto (en inglés Pareto-envelope Based Selection Algorithm, PESA) que combinaba aspectos de PAES [KNO99], y SPEA [ZIT99]. Esta aproximación utiliza una pequeña población interna y un gran archivo externo (o población secundaria) donde se almacena la población. Con el objetivo de mantener la diversidad, PESA usa la estrategia de almacenamiento de malla adaptativa (o hiper-malla) que utiliza PAES [KNO99], en combinación con el usado en el procedimiento de crowding para establecer la selección de individuos, persiguiendo que las soluciones se distribuyan uniformemente en la hiper-malla. Este mismo procedimiento se utiliza para decidir qué soluciones se introducen en el archivo externo de población [KNO04].

Pareto-envelope Based Selection Algorithm II (PESAII)

Tras analizar algunas posibles mejoras sobre PESA, Corne y col. propusieron PESAII [COR01]. Esta mejora incluía, entre otros aspectos, una adaptación de la estrategia de selección basada en la hiper-malla, conocida como estrategia basada en region (*region-based strategy*). En dicha estrategia, la unidad mínima de selección es un hiper-cubo que contiene más de un individuo. El procedimiento consiste en seleccionar una región de la hiper-malla, posteriormente seleccionar un individuo de ella, haciendo uso de alguno de los procedimientos de selección existentes. El resto del procedimiento es muy similar a PESA.

Micro Genetic Algorithm (micro-GA)

El término micro algoritmo genético (en inglés micro Genetic Algorithm, micro-GA) se refiere a un algoritmo genético con una población muy pequeña y un proceso de reinicialización. La idea fue sugerida por algunos resultados teóricos obtenidos por Goldberg [GOL89], de acuerdo a las cuales una población de tres individuos era suficiente para converger sin verse influenciado por la longitud del cromosoma. El proceso sugerido por Goldberg fue el de aplicar operadores genéticos a una población pequeña (generada aleatoriamente), hasta alcanzar convergencia nominal (es decir, cuando todos los individuos tienen sus genotipos idénticos o muy similares). Posteriormente, se genera una nueva población mediante la transferencia de los mejores individuos de la población anterior, mientras que los restantes individuos son generados aleatoriamente.

El primer informe de una implementación de un micro-GA fue realizado por Krishnakumar [KRI89], quien usó una población de tamaño 5, un porcentaje de cruce del 100 % y un porcentaje de mutación del 0 %. Dicho método también adoptó una estrategia de elitismo, de forma que la mejor cadena encontrada en la población actual pasaba directamente a la siguiente generación. La selección fue desempeñada por el sostenimiento de 4 competidores entre soluciones que eran adyacentes en el arreglo poblacional, y declarando como ganador al individuo con la más alta aptitud. Krishnakumar [KRI89] comparó su micro-GA frente a un GA simple, cuyos parámetros eran P_{tam}=50, ρ_{cruce}=60 %, ρ_{mut}=0.01. Además de ser más rápido, este micro-GA obtenía mejores resultados sobre dos funciones estacionarias, y un problema de control del mundo real.

Posteriormente, Toscano y Coello propusieron el llamado micro-GA2 [TSC03]. El micro-GA2 usa un mecanismo de adaptación en línea que hace innecesario el uso de estos parámetros. Incluso puede decidir cuando parar (no dispone de un parámetro que le indique el número mínimo de generaciones). El único parámetro que se requiere es el tamaño del archivo de almacenamiento externo.

2.2.2. Algoritmos Multi-objetivo Basados en Búsqueda Local

Multi-Objective Simulated Annealing de Serafini (SMOSA)

Como describimos en el Capítulo 1, el enfriamiento simulado (SA) [KIR83] es una técnica de relajación estocástica basada en la analogía al proceso físico de enfriamiento de un metal. En el contexto mono-objetivo, SA acepta todas

las soluciones vecinas que mejoran a las anteriores, mientras que las que empeoran se aceptan con una probabilidad determinada por el criterio de Metrópolis [MET53], que a su vez depende del valor actual de la *temperatura*. En el contexto multi-objetivo hay que considerar un nuevo caso cuando ambas soluciones, la inicial y la vecina obtenida a partir de ella son indiferentes. La primera adaptación multi-objetivo de SA fue el enfriamiento simulado multi-objetivo de Serafini (en inglés Serafini's Multi-Objective Simulated Annealing, SMOSA) [SER93]. Esta técnica utiliza una única solución durante el proceso de búsqueda, a la vez que se establecen varias probabilidades de transición utilizando funciones escalares de suma de pesos, en conjunción con relaciones de Pareto-dominancia. Al ser un método basado en trayectorias, el frente óptimo se va generando dinámicamente mediante el almacenamiento de las soluciones no dominadas encontradas.

Multi-Objective Simulated Annealing de Ulungu (UMOSA)

Con posterioridad a SMOSA, Ulungu presentó otro algoritmo multi-objetivo basado en enfriamiento simulado (en inglés Multi-Objective Simulated Annealing de Ulungu, UMOSA) [ULU99] que utiliza funciones escalares durante el proceso de búsqueda, de forma que se llevan a cabo diferentes ejecuciones con diferentes pesos en cada objetivo (ver Ecuación 2.1). De esta forma en algunas ejecuciones se consideran más importantes ciertos objetivos que otros, haciendo uso de un parámetro σ_k, que de forma similar a las funciones de agregación, guía el proceso de búsqueda. Al finalizar todas las ejecuciones, se unen las soluciones no dominadas obtenidas tras cada una de ellas. El uso de un conjunto lo suficientemente diverso de pesos permite guiar el proceso de búsqueda en varias direcciones, en función del objetivo que se desea minimizar. En sus conclusiones, Ulungu sugiere como investigación futura, la comparación de su estrategia frente a otras como [CZY98], aspecto que tratamos en esta tesis.

Pareto Simulated Annealing de Czyzak (PSA)

Como acabamos de ver, tanto SMOSA como UMOSA hacen uso de una sola solución durante el proceso de búsqueda. Sin embargo, otros autores han planteado algoritmos basados en enfriamiento simulado que hacen uso de poblaciones. En concreto, Czyzak y Jaszkiewicz presentaron el método de enfriamiento simulado de Pareto (en inglés Pareto Simulated Annealing, PSA) [CZY98]. PSA utiliza una población principal y un archivo externo donde se almacenan las mejores soluciones encontradas en dicha población principal. Además, PSA controla

dinámicamente los pesos de la función objetivo, con lo que se intenta distribuir las soluciones a lo largo de los diferentes objetivos de forma homogénea. Los resultados de PSA son comparados frente a SMOSA en una formulación multi-objetivo del problema de la mochila [CZY98]. Sin embargo, no se compara con MOMHs basadas en otras técnicas de búsqueda local, aspecto que es tratado en esta tesis.

Hansen's Multi-Objective Tabu Search (MOTS)

Al igual que el enfriamiento simulado, la búsqueda tabú también ha sido utilizada en el contexto multi-objetivo, destacando la búsqueda tabú multi-objetivo (en inglés Multi-Objective Tabu Search, MOTS) [HAN97]. MOTS es una técnica poblacional, cuyas soluciones son mejoradas mediante el uso de un criterio de aceptación basado en TS. En MOTS, los pesos de cada objetivo son adaptados de forma dinámica, al igual que en PSA. Aunque los experimentos [HAN97] evalúan la variación en la calidad de las soluciones de acuerdo a la longitud de la lista tabú, no se ofrece comparativa con otros métodos, por lo que las comparativas incluidas en esta tesis responden a ello.

The Pareto Archived Evolution Strategy (PAES)

La estrategia de evolución con archivo de Pareto (en inglés Pareto Archived Evolution Strategy, PAES) [KNO99] fue otra técnica de búsqueda que ofrecía, entre otras, las siguientes características básicas:

- utiliza un solo punto para la búsqueda local;
- utiliza selección basada en ranking/dominancia de Pareto;
- devuelve un número limitado de soluciones no dominadas;
- utiliza un archivo externo para almacenar soluciones no dominadas basadas en una malla de regiones (hiper-malla);
- trabaja con un número de parámetros reducido.

Desafortunadamente, en la práctica se presentan varios conflictos entre estas características. En particular, destaca el hecho de que utilizar una sola solución durante el proceso de búsqueda resulta insuficiente para encontrar un conjunto extenso de soluciones no dominadas cuando el tamaño del problema es elevado. Todos estos aspectos son analizados detalladamente en [KNO99].

2.2.3. Algoritmos Híbridos Multi-objetivo

La hibridación de meta-heurísticas permite la cooperación de métodos teniendo características complementarias. La hibridación presenta una serie de ventajas, tales como su habilidad para obtener soluciones próximas al óptimo de Pareto, principalmente cuando hay un gran número de óptimos locales. El interés en el diseño e implementación de meta-heurísticas híbridas se ha incrementado de forma importante en los últimos años [TAL02]. Sin embargo, la mayoría de estos estudios se han centrado en técnicas mono-objetivo, mientras que en el ámbito multi-objetivo constituye un área de investigación aun muy abierta. A continuación describimos brevemente algunas meta-heurísticas híbridas para optimización multi-objetivo (hMOMHs) encontradas en la literatura.

La búsqueda local genética multi-objetivo (en inglés Multi-Objective Genetic Local Search, MOGLS) [ISH98] es un algoritmo genético para optimización multi-objetivo que adicionalmente aplica búsqueda local en cada individuo de la población haciendo uso de una función objetivo de suma de pesos. Los resultados obtenidos por MOGLS en problemas de planificación de tareas resultaron muy satisfactorios [ISH98].

Knowles y Corne extendieron PAES haciendo uso de algoritmos meméticos, obteniendo como resultado el PAES memético (en inglés memetic-PAES, M-PAES) [KNO00]. M-PAES hace uso de una población de soluciones y un operador de cruce que es utilizado para recombinar soluciones encontradas por PAES. Al igual que en PAES, las soluciones prometedoras en la búsqueda se almacenan en un archivo externo de soluciones no dominadas (ND). Los resultados obtenidos en el problema de la mochila [KNO00] denotaron el buen comportamiento de M-PAES con respecto a SPEA [ZIT99].

Hu y col. [HU03] combinaron la programación cuadrática secuencial (en inglés Sequential Quadratic Programming, SQP) con SPEA [ZIT99] y NSGA-II [DEB00]. Esta hibridación permite una mejora en la calidad de las soluciones en diferentes funciones matemáticas de prueba. Además, esta estrategia híbrida permite incrementar la diversidad de las soluciones obtenidas.

Otra hMOMH recientemente propuesta es la búsqueda genética tabú (en inglés Genetic Tabu Search, GTS) [BAR03]. GTS es un método híbrido que combina la naturaleza global de los algoritmos genéticos con la naturaleza local de la búsqueda tabú. Los resultados obtenidos en el problema de la mochila demostraron que GTS igualaba y mejoraba los resultados obtenidos por NSGA y MOGLS [BAR03].

2.3. Conclusiones

La mayoría de los problemas de optimización conllevan la maximización y/o minimización de varios objetivos de forma simultánea. En la mayoría de los casos esos objetivos suelen estar en conflicto, es decir, la mejora en algunos de ellos provoca el deterioro en otros. En optimización mono-objetivo existe un óptimo global, mientras que en el contexto multi-objetivo no existe una solución claramente definida sino un conjunto que optimiza todos los objetivos. En la última década, la mayoría de los trabajos en el campo multi-objetivo hacen uso del concepto de Pareto-optimalidad. El objetivo de las estrategias basadas en frentes de Pareto es generar un conjunto de soluciones no dominadas como una aproximación al frente Pareto-óptimo.

En este capítulo hemos presentado los conceptos necesarios parar tratar problemas multi-objetivo haciendo uso del concepto de optimización basada en frentes de Pareto. Así mismo, se han descrito las principales técnicas heurísticas multi-objetivo encontradas en la literatura. Aunque durante los últimos años se ha incrementado notablemente el esfuerzo de investigación relacionado con el diseño de técnicas heurísticas multi-objetivo, en el ámbito de combinar técnicas aun es necesario realizar más esfuerzos de investigación.

Los conceptos presentados en este capítulo son de gran utilidad a la hora de comprender las formulaciones multi-objetivo de los problemas descritos en el Capítulo 3, así como el fundamento de los procedimientos híbridos multi-objetivo presentados en el Capítulo 5, y de una de las paralelizaciones descritas en el Capítulo 6.

Capítulo 3

Problemas de Optimización Analizados

En el presente capítulo se ofrece una descripción detallada de los problemas de optimización tratados en esta tesis doctoral. En concreto se analizan dos problemas de optimización de gran complejidad y con un elevado número de aplicaciones prácticas, como son el problema de la repartición de grafos, y el diseño de redes de distribución de agua malladas.

Ambos problemas resultan bastante adecuados para evaluar el comportamiento de las heurísticas de optimización. Su elección ha venido motivada por tres razones principales. La primera es que se trata de problemas combinatorios de gran complejidad (el problema de repartición de grafos es NP-completo [GAR79], mientras que el problema de redes de distribución de agua es NP-duro [GUP93]). En segundo lugar, cabe indicar que hasta la fecha se han realizado relativamente pocos estudios haciendo uso de formulaciones multi-objetivo basadas en optimización de Pareto en ambos problemas. En tercer lugar, ambos problemas resultan de especial interés dado su gran número de aplicaciones prácticas. Por un lado el problema de repartición de grafos se puede extender a múltiples aplicaciones reales, como repartición de circuitos, repartición de mallas, etc., mientras que el diseño de redes de distribución de agua abarca una gran cantidad de sistemas, como suministro urbano, redes de riego, etc.

La descripción de dichos problemas en este capítulo permite que, una vez descritos ambos problemas en sus formulaciones mono-objetivo y multi-objetivo, las instancias de prueba utilizadas, y las métricas multi-objetivo utilizadas, resultará inmediato el análisis de los resultados obtenidos por los procedimientos heurísticos propuestos en capítulos posteriores.

3.1. Problema de Repartición de Grafos (GPP)

La repartición de grafos es un importante problema de optimización incluido dentro de la categoría de complejidad NP-completo [GAR79], lo cual implica que para problemas de tamaño realista es necesario aplicar procedimientos eficientes que ofrezcan soluciones de calidad en tiempos razonables. Como ejemplos de aplicaciones de dicho problema en disciplinas de ciencia e ingeniería encontramos el diseño de circuitos integrados de gran escala (VLSI) [ALT95, GIL98, JIA03], aplicaciones de mecánica computacional [HEN00], procesamiento de matrices dispersas [KAR98b], distribución equilibrada de carga en computadores [WAL02, DEV05], planificación de tareas [ALE01, ERC05], minería de datos [ZHA01, BEK02], almacenamiento eficiente de grandes bases de datos en discos [SHE96, KOY05], etc. En particular, los modelos de distribución de datos en multiprocesadores de memoria distribuida tratan de distribuir equitativamente la carga computacional entre los diferentes procesadores a la vez que se minimiza el volumen de comunicaciones entre los mismos [DEV05]. Otra aplicación real donde la eficiencia de los algoritmos de repartición es crítica lo encontramos en aplicaciones de mecánica computacional, donde los elementos y volúmenes finitos son representados mediante mallas. Una malla consiste en un conjunto de elementos que representan la discretización de un dominio geométrico [BEN00]. Las mallas suelen ser clasificadas en dos categorías: estructuradas y no estructuradas. Las mallas estructuradas se caracterizan por su conectividad regular, es decir, los elementos de las mallas estructuradas tienen la topología de una parrilla regular. Por el contrario, una malla no estructurada se caracteriza por su conectividad irregular, cuyos elementos en el espacio bi-dimensional suelen ser triángulos o cuadriláteros, mientras que en tres dimensiones suelen ser tetrahedros y hexahedros. Las mallas no estructuradas suelen ser más difíciles de computar que las estructuradas, pero tienen la ventaja de que sus tamaños pueden variar en función de la aplicación, lo cual permite obtener mejores representaciones de ciertas estructuras que en el caso de utilizar mallas estructuradas.

Trabajar con mallas que representan grandes estructuras requiere de importantes recursos de procesamiento y memoria. Una de las formas más eficientes de tratar este inconveniente es distribuir la malla entre diferentes procesadores a la vez que se satisfacen dos condiciones. La primera es que el número de elementos asignados a cada procesador sea el mismo, es decir, que las computaciones entre procesadores estén equilibradas. El segundo requisito es que el número de elementos contiguos asignados a diferentes procesadores sea mínimo, es decir,

que se reduzca el volumen de comunicaciones entre diferentes procesadores. Una forma interesante de llevar a cabo esta tarea es representar la malla mediante un grafo dual, un grafo nodal, o una combinación de ambos. Los vértices de un grafo dual representan los elementos finitos, mientras que las aristas entre dos vértices indican que los elementos correspondientes son adyacentes. Los vértices de un grafo nodal representan los nodos de la malla. La Figura 3.1(a) muestra una malla bi-dimensional no estructurada con 14 elementos. La Figura 3.1(b) muestra el grafo dual asociado a la malla, donde los vértices representan elementos, y las aristas entre dos vértices indican que dos elementos son adyacentes. La Figura 3.1(c) muestra una repartición del grafo dual, mientras que la Figura 3.1(d) muestra la malla asociada ya dividida, la cual ha obtenido dos sub-mallas de siete elementos cada una con cuatro lados comunes entre elementos pertenecientes a diferentes sub-mallas.

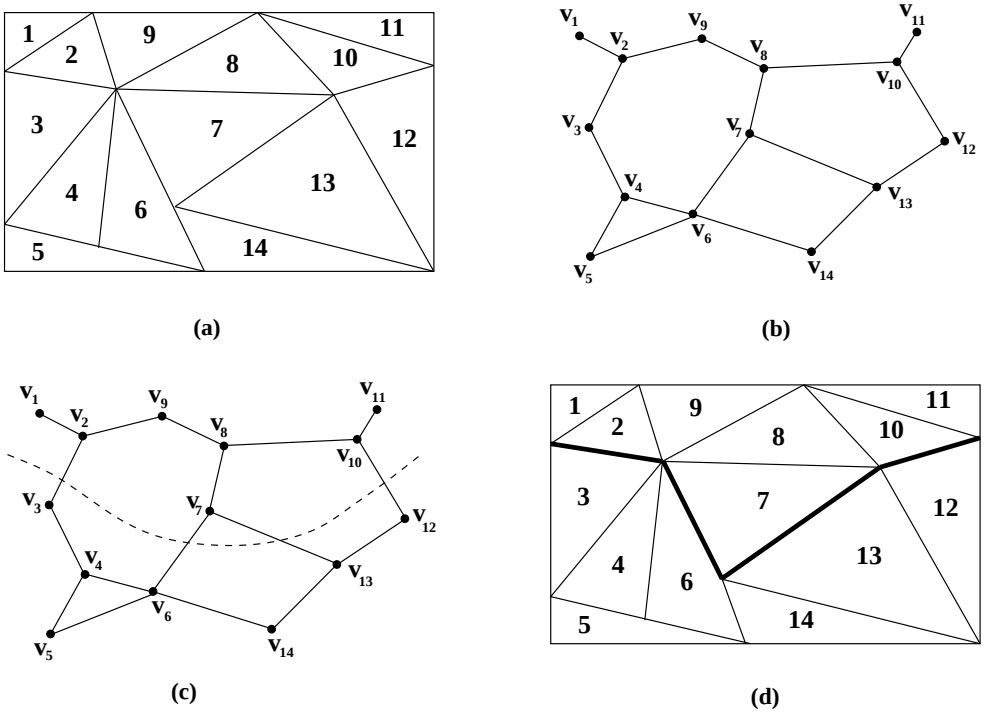

Figura 3.1: (a) Malla, (b) grafo dual asociado a la malla, (c) repartición del grafo dual, (d) traslación a la malla.

Otro ejemplo de aplicación de repartición de grafos lo encontramos en las redes de comunicaciones [TOL96, WAL01, CHE04]. La Figura 3.2(a) muestra una pequeña red celular y su grafo dual asociado. Las líneas discontinuas muestran

las celdas adyacentes. La Figura 3.2(b) muestra una posible repartición del grafo dual asociado, cuya equivalencia en la red celular viene descrita en la Figura 3.2(c).

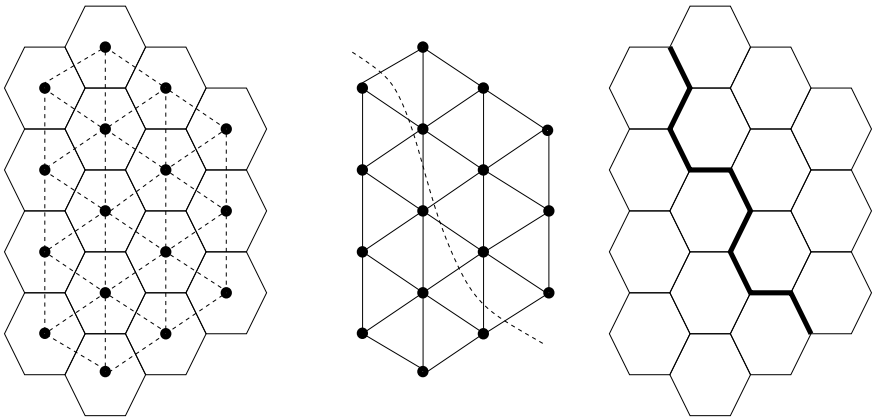

Figura 3.2: (a) Red celular, (b) repartición del grafo dual asociado, (c) repartición equivalente en la red celular.

3.1.1. Descripción del Problema

En la formulación mono-objetivo del problema de repartición de grafos (GPP) [GPA] se persigue encontrar aquella partición de un grafo cuyo número de aristas pertenecientes a diferentes sub-grafos sea mínimo, cumpliendo a su vez que la suma de pesos de los vértices de cada sub-grafo esté equlibrada. A continuación se ofrece una descripción matemática de dicha formulación.

Definición 3.1. Problema de Repartición de Grafos. Dado un grafo G=(V,E) no dirigido, con pesos, sin auto-aristas, o múltiples aristas entre dos vértices, donde V es el conjunto de vértices $V = \{v_1, v_2, ..., v_n\}$, y E el conjunto de aristas tal que $E = \{e_{i,j} \mid$ existe una arista entre v_i y $v_j\}$. El GPP consiste en dividir V en SG sub-grafos equilibrados, V_1,..., V_{SG}, tal que la suma de pesos entre aristas conectando vértices de diferentes sub-grafos sea minimo, verificando que: $V_i \cap V_j = \phi$, $\forall\ i \neq j$; $\cup_{sg=1}^{SG} V_{sg} = V$; $W_{Vsg} \approx W_V/SG$, donde W_V representa la suma de pesos de los vértices del grafo y W_{Vsg} representa la suma de pesos de los vértices del sub-grafo *sg*.

Por lo tanto, esta formulación intenta encontrar la solución cuya suma de

pesos de aristas que conecten vértices de diferentes sub-grafos (cortes) sea mínima, a la vez que el peso total de vértices entre sub-grafos se mantiene equilibrado. Sin embargo, debido a la alta dificultad de reparticionar grandes grafos, es habitual suavizar las restricciones de desequilibrio. En muchos casos se admiten particiones cuyo desequilibrio se encuentre por debajo de un determinado umbral. Dicho desequilibrio viene definido por el peso del sub-grafo más grande, M=max(W_i), $\forall i \in [1,..,SG]$. De esta forma, si el desequilibrio máximo permitido es DB_{max} %, entonces la partición obtenida deberá verificar que $M \leq ((W_V/SG)*((100+DB_{max})/100))$. El principal inconveniente de este modelo es la alta dependencia de la restricción de desequilibrio. Así, por ejemplo, la Figura 3.3(a) muestra la partición de un grafo totalmente equilibrada, aunque el número de cortes no es mínimo. Por otro lado, la Figura 3.3(b) muestra una partición con el número mínimo de cortes, pero desequilibrada. Bajo estas circunstancias, el seleccionar una u otra solución estaría en función del grado de desequilibrio máximo permitido (DB_{max} %).

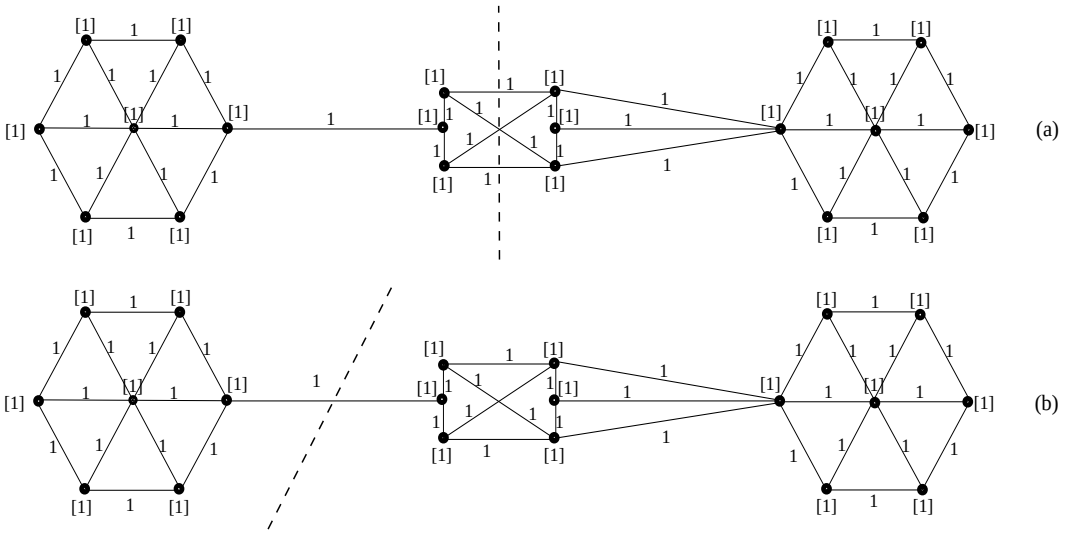

Figura 3.3: Dos formas de dividir un grafo.

Métodos de Repartición de Grafos

Desafortunadamente, incluso en el caso más simple donde los vértices y aristas no tienen pesos (es decir, tienen peso 1) y SG=2, el GPP sigue perteneciendo a la categoría de complejidad NP-completo [GAR79]. Hasta el momento no exis-

te, y es probable que no se pueda diseñar un algoritmo eficiente para resolver este problema, por lo que resulta de gran utilidad el diseño de procedimientos que obtengan soluciones de calidad aceptable en tiempos de ejecución admisibles. Como resultado del gran interés en el diseño de algoritmos de repartición de grafos durante las últimas décadas se han propuesto un gran número de métodos, que pueden ser clasificados en las diferentes categorías, tal y como describimos a continuación.

- Métodos *globales* vs. *locales*. Algunas técnicas toman una visión global del grafo, ya que consideran toda su estructura a la hora de llevar a cabo la repartición [SIM91]. Sin embargo, los llamados métodos locales [KER70] trabajan con una visión limitada del problema, considerando una partición de entrada ya procesada, la cual es mejorada obteniendo una partición de más calidad.

- Métodos *geometricos* vs. *libres de coordenadas*. Los métodos geométricos [GLB98] llevan a cabo la repartición considerando la localización espacial de los vértices con el objetivo de asignar vértices contiguos al mismo subgrafo. También existe otra clase de métodos libres de coordenadas [SIM91], que intentan reducir el número de cortes haciendo uso de la información sobre conectividad entre vértices.

- Métodos *estáticos* vs. *dinámicos*. La mayoría de los métodos de repartición se consideran estáticos [SIM91] debido a que han sido diseñados para obtener soluciones de gran calidad, suponiendo que la estructura del grafo es invariable durante toda la ejecución. Sin embargo, existen otras aplicaciones como el de equilibrado de carga, donde la estructura de datos evoluciona en el tiempo, con lo que resulta incluso más importante la velocidad en obtener la partición que su calidad. Bajo dichas circunstancias, los algoritmos dinámicos [SCH01] son más recomendables.

- Métodos *mono-nivel* vs. *multi-nivel*. Los algoritmos clásicos [BEG87] directamente dividen el grafo de entrada. Sin embargo, en algunas aplicaciones los grafos son tan extremadamente grandes, que resulta inviable realizar dicha partición. En la última década, la mayoría de los algoritmos de repartición global libres de coordenadas han hecho uso del paradigma multi-nivel [HEN93], que tiene la ventaja de computar buenas particiones en tiempos más reducidos que los métodos mono-nivel.

- Métodos *secuenciales* vs. *paralelos*. La mayoría de los algoritmos de repartición se han diseñado para la ejecución en un solo procesador [KAR98a], lo cual puede no resultar factible para muchas aplicaciones dinámicas, o incluso estáticas cuando el tamaño del problema a resolver es muy elevado. El incremento en la velocidad y capacidad de memoria de máquinas paralelas ha permitido que el procesamiento paralelo y distribuido se haya convertido en una interesante, si no la mejor, herramienta para llevar a cabo dichas computaciones [KAR98b].

La elección de una determinada técnica dependerá de la aplicación en cuestión. Así, para aplicaciones estáticas, como el diseño VLSI [JIA03], el objetivo es obtener la mejor solución posible independientemente del tiempo requerido para su obtención. Sin embargo, para aplicaciones dinámicas, como la distribución equilibrada de carga en computadores [DEV05], no resulta tan importante obtener la mejor solución posible, sino obtener una solución aceptable en un tiempo reducido. Algunos autores han analizado el rendimiento de varios algoritmos de repartición estática y dinámica [ELS02]. A continuación ofrecemos una descripción de los principales algoritmos de repartición encontrados en la literatura, clasificándolos en función de si hacen uso de información de la geometría del grafo o de su conectividad.

Métodos Geométricos

Las técnicas geométricas [GLB98] son aproximaciones clásicas en las cuales la repartición se lleva a cabo en función de la localización espacial de los vértices dada por sus coordenadas. En el caso de las mallas, las coordenadas a considerar suelen ser las de los nodos de los elementos o bien sus centroides. Esas técnicas tienden a agrupar en la misma sub-malla a aquellos elementos cercanos físicamente. Los métodos geométricos son muy rápidos y ofrecen buenas soluciones para aplicaciones que necesitan obtener particiones equilibradas cuando se conoce que los vértices están distribuidos de forma homogénea. Otra ventaja es que los métodos geométricos son inherentemente paralelos. Sin embargo, los métodos geométricos, al no considerar la información de conectividad del grafo, suelen obtener particiones con un número de cortes elevado. A continuación se describen los métodos geométricos más conocidos en la literatura.

La bisección de coordenadas recursiva (en inglés Recursive Coordinate Bisection, RCB) [BEG87], crea particiones dividiendo recursivamente un grafo dado en dos sub-grafos del mismo peso hasta que el número de sub-grafos obtenidos

alcanza el número deseado. Este método determina los sub-grafos tomando como referencia un plano ortogonal a un eje de coordenadas, tal y como muestra la Figura 3.4(a). La dirección del plano se selecciona ortogonal a la dirección más larga de la geometría, y su posición es aquella en la que la mitad de vértices se localizan en cada lado del plano. El esquema RCB es extremadamente rápido, requiere poca memoria, y es fácilmente paralelizable, además de válido para equilibrio dinámico de carga [HEN00]. Sin embargo, la calidad de las soluciones suele ser pobre cuando se reparticionan estructuras altamente irregulares.

La bisección inercial recursiva (en inglés Recursive Inertial Bisection, RIB) [NOU87], es una generalización de RCB que también corta el plano mediante su bisección recursiva, pero de una forma diferente. En concreto, se computa el eje principal de esta estructura el cual corresponde a la dirección en la cual el grafo es alargado. Los vértices se dividen en sub-grafos de pesos similares ortogonales al eje principal. Dicho procedimiento se repite de forma recursiva para cada sub-grafo generado, tal y como se muestra en la Figura 3.4(b). Al igual que RCB, RIB también es muy rápido, y aunque obtiene particiones de baja calidad en términos de número de cortes.

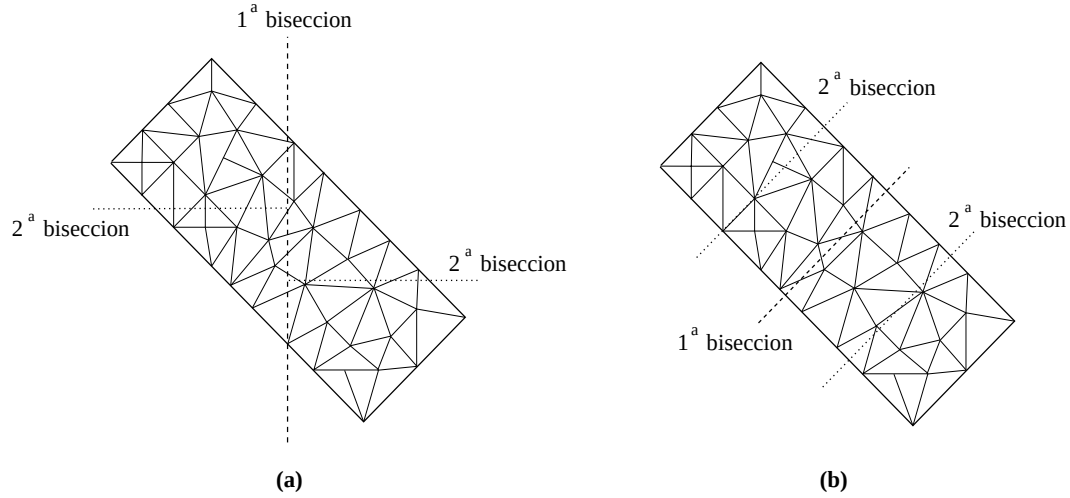

Figura 3.4: (a) Bisección de Coordenadas Recursiva (RCB), (b) Bisección Inercial Recursiva (RIB).

Las curvas de relleno del espacio (en inglés Space-filling Curves, SFC) [SCG05] son curvas continuas que rellenan espacios de dimensiones más grandes tales como cuadrados o cubos. En RCB y RIB los elementos son calculados mediante

la proyección de los elementos en ejes de coordenadas e inerciales, es decir, dichas dichas computaciones se realizan teniendo en cuenta de una sola dimensión. Sin embargo, el uso de más dimensiones pueden ofrecer mejores particiones, tal y como se demostró haciendo uso de SFCs. Las SFCs producen particiones con la característica deseada de que elementos cercanos entre sí en el espacio tienen más probabilidades de ser agrupadas el mismo sub-grafo. El uso de SFCs suele ofrecer buenas y rápidas particiones, en la mayoría de los casos algo mejores que RCB y RIB. Algunas SFCs son las curvas de Peano, Peano-Hilbert, Sierpinski, Dragon, etc. Más información acerca de SFCs se puede encontrar en [HAS94].

La bisección circular (en inglés Circle Bisection, CB) [MIL93] es un algoritmo recursivo que solventa muchos de los inconvenientes de los algoritmos basados en hiper-planos. CB calcula los separadores haciendo uso de círculos en el espacio bi-dimensional y esferas en tres dimensiones, en lugar de utilizar de líneas y planos, respectivamente. Debido a que se puede utilizar un radio variable para realizar la partición, el grado de libertad es mayor y por tanto la calidad de las soluciones tiende a mejorar.

Métodos Libres de Coordenadas

La principal desventaja de métodos geométricos radica en que no consideran la conectividad entre vértices, lo que provoca que, en la mayoría de los casos se obtengan particiones equilibradas pero con un elevado número de cortes. Existen otros métodos libres de coordenadas, también llamados métodos combinatorios, que no consideran la localización espacial de los nodos, sino la conectividad entre los mismos. Aunque los métodos geométricos son más rápidos que los libres de coordenadas, estos últimos tienden a obtener particiones con un menor número de cortes. A continuación se describen los principales métodos libres de coordenadas encontrados en la literatura, clasificados en dos sub-categorías: locales y globales.

Métodos locales. Un procedimiento de división local obtiene particiones a partir de un grafo dado, G, tomando como entrada una partición previamente obtenida por otros procedimientos. Mientras otros métodos de repartición ofrecen una visión global del problema, los métodos de mejora local consideran las características particulares del grafo. Los métodos locales mejoran una partición inicial obtenida aleatoriamente o mediante alguna técnica determinista, razón por la cual se denominan heurísticas de mejora. Uno de los primeros procedimientos de repartición de grafos fue el propuesto por Kernighan y Lin (KL)

[KER70]. KL fue propuesto originariamente para optimizar el emplazamiento de circuitos electrónicos en tarjetas de forma que las conexiones entre tarjetas fuera mínimo. El funcionamiento general de KL está basado en tomar un grafo dado G dividido previamente en dos sub-grafos (V_1, V_2), y mejorar la calidad de dicha partición mediante el intercambio de pares de vértices (v_i, v_j) pertenecientes a diferentes sub-grafos con el objetivo de minimizar el número de cortes, mientras que el desequilibrio permanece constante. Con el objetivo de describir formalmente el algoritmo KL, se necesitan definir algunos conceptos, tal y como se muestra a continuación.

$$int(v_i) = \Sigma_{(i,j)\in E, Sub(v_i)=Sub(v_j)} w(v_i, v_j) \tag{3.1}$$

$$ext(v_i) = \Sigma_{(i,j)\in E, Sub(v_i)\neq Sub(v_j)} w(v_i, v_j) \tag{3.2}$$

$$gain(v_i) = ext(v_i) - int(v_i) \tag{3.3}$$

$$gain(v_i, v_j) = gain(v_i) + gain(v_j) - 2w(v_i, v_j) \tag{3.4}$$

Cada iteración de KL comienza marcando como no visitados todos los vértices. Entonces, un par de vértices no marcados $v_1 \in V_1$ y $v_2 \in V_2$ para los cuales gain(v_1,v_2) es máximo, son marcados v_1 y v_2, y se actualizan los valores de ganancia para todos los vecinos de v_1 y v_2, como si dicho intercambio se hubiera realizado. Una vez se evaluan las ganancias de los vértices analizados, se puede construir una lista ordenada de parejas de vértices (v_1^i, v_2^i), i=1,..,n. El siguiente paso consiste en encontrar el índice j tal que $\Sigma_{i=1}^{j}$ gain(v_1^i,v_2^i) es máximo. Si esta suma es mayor que cero, los primeros j pares de vértices son intercambiados, y se lleva a cabo otra iteración. En otro caso, el algoritmo finaliza.

Algunos años más tarde, Fiduccia y Mattheyses propusieron una variante de KL, denominada FM [FID82]. Como KL, FM mueve vértices entre sub-grafos vecinos. Sin embargo, mientras KL intercambia pares de vértices, FM mueve un solo vértice en cada paso, siendo este el que ofrece valores de ganancia son máximos, mientras que el desquilibrio se mantiene por debajo de un umbral previamente establecido. La principal ventaja de KL y FM radica en su rapidez y fácil implementación. KL y FM son métodos incluidos dentro de los llamados algoritmos de ascenso/descenso de colinas. Dichos algoritmos siempre paran en el primer óptimo local encontrado, es decir, FM detiene la búsqueda cuando la

suma de las ganancias no es mayor que cero. Este factor implica que la calidad de las particiones generadas por dichos algoritmos dependen fuertemente de la calidad de la partición inicial utilizada. Si esas particiones iniciales son de baja calidad, KL y FM suelen ser incapaces de mejorarlas. Por otro lado, KL y FM fueron diseñadas para bi-reparticionar grafos sin pesos. Sin embargo, otros autores han generalizado los algoritmos KL y FM para un número arbitrario de sub-grafos con pesos [HEN95, LAR06].

La optimización de rutas (en inglés Path Optimization, PO) [BER95] se puede entender como una variación de la búsqueda local basada en descenso de colinas. Mientras que KL y FM mueven un número reducido y constante de vértices, PO desarrolla secuencias de movimientos de vértices vecinos de longitud variable. La principal ventaja de este método radica en examinar un gran número de vértices antes de llevar a cabo un determinado movimiento. Berry and Goldberg [BER95] evaluaron el rendimiento de PO en el GPP, comprobando que trabajaban correctamente en comparación con KL.

En un intento de resolver las desventajas de las técnicas de descenso de colinas, en años recientes algunos autores han propuesto procedimientos heurísticos que evitan quedar atrapados en mínimos locales con el objetivo de alcanzar mejores soluciones. Una de esas técnicas es el enfriamiento simulado (SA) [KIR83], descrita en el Capítulo 1. Johnson y col. [JOH89] aplicaron SA para la bisección de grafos y lo compararon experimentalmente con KL de forma que SA mejoraba a KL en la mayoría de los casos.

La búsqueda tabú (TS) [GLO97] ha sido también adaptada al GPP. Así, por ejemplo, Rolland y col. [ROL96] utilizaron TS para bisección de grafos de forma que los vértices que son intercambiados de un sub-grafo a otro en el vecindario son incluidos en la lista tabú durante un número determinado de iteraciones. Si no hay mejora tras varias iteraciones, la restricción de desequilibrio es incrementada. Los resultados demostraron que TS mejoraba el rendimiento obtenido por KL y SA en términos de calidad de soluciones, además de requerir tiempos de ejecución reducidos.

Los algoritmos evolutivos (EA) [FOG06] también han sido aplicados a problemas de partición. Así, Gil y col. [GIL98] combinaron un EA con SA y TS en el problema de repartición de circuitos. Recientemente, Soper y col. [SOP04] mejoraron el rendimiento del operador de cruce haciendo uso del paradigma multi-nivel [HEN93].

Los algoritmos meméticos (MA) [MOS00] combinan aspectos de otras heurísticas, principalmente de algoritmos evolutivos y búsqueda local. Algunos autores

han presentado MAs para resolver el GPP. Merz y Freisleben [MER00] propusieron un MA para reparticionar grafos. Su comparación entre diferentes variantes de MA con el método Diff-Greedy multiarranque [BAT97] combinado con KL (DG+KL) y también con un algoritmo iterativo KL (IKL). Los resultados demostraron que los MAs mejoran a esos otros métodos.

Una de las técnicas heurísticas con más expansión en los últimos años ha sido la optimización mediante colonias de hormigas (ACO) [DOR99]. Korosec y col. [KOR04] evaluaron el rendimiento de ACO en comparación con diferentes métodos de partición como Metis [MES], Chaco [CHA94], y MLSATS [BAÑ03] en el problema de repartición de mallas. Los resultados obtenidos en dicho problema indicaron que ACO funciona muy bien en grafos de tamaño pequeño y medio, mientras que en grafos más grandes es necesario aplicarlo en combinación con el paradigma multi-nivel para obtener soluciones de calidad. Recientemente, Comellas y Sapena [COM06] han propuesto un algoritmo basado en multi-agentes (*ants*) que alcanza y mejora a otros métodos propuestos con anterioridad.

Una red neuronal (NN) [WAS89] es un sistema para interconectar neuronas en una red trabajando conjuntamente para producir una función de salida. La red neuronal permite la cooperación de neuronas individuales dentro de la red a evaluar. Las redes neuronales han sido también aplicadas para resolver el GPP [SHI99, MED05]. Así, por ejemplo, Mérida y López [MED05] aplicaron un modelo unificado basado en una técnica neuronal para GPP. Los resultados obtenidos comparados con otras alternativas, demostraron que es un algoritmo eficiente y competitivo en términos de calidad de soluciones y tiempo computacional.

Métodos globales. Los métodos globales se suelen denominar heurísticas constructivas, ya que toman la descripción del grafo como entrada y generan una primera partición a partir de la misma.

La bisección espectral (en inglés Spectral Bisection, SB) [POT90] es una técnica para bisección de grafos que suele producir mejores particiones para una amplia variedad de problemas [HEN95, KEN06]. Es un método global ya que la partición se lleva a cabo considerando el grafo en conjunto. El primer paso de SB consiste en determinar la matriz Laplaciana del grafo, $L(G)$, tal que $L(i,j)=-1$ si los vértices i y j son adyacentes, $i \neq j$, y $L(i,i)$ representa el número total de conexiones entre i y cualquier otro nodo. $L(G)$ es simétrico y sus filas suman cero. Una vez se calcula $L(G)$, el siguiente paso de SB es calcular el segundo autovalor más pequeño, también conocido como vector Fielder $\lambda_2(L(G))$, que representa la conectividad del grafo G. Así, el vector v_2 correspondiente a $\lambda_2(L(G))$ es el

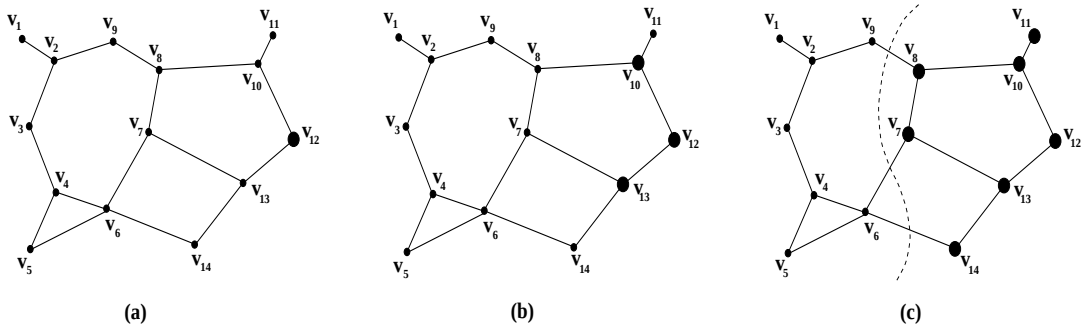

Figura 3.5: Algoritmos de crecimiento (growing) para repartición de grafos.

autovalor de conectividad algebraica, el cual tiene propiedades especiales. Una vez calculado el vector Fielder, el grafo G es biseccionado de forma que el vértice n es asignado al primer sub-grafo si $v_2(n)<0$, en otro caso este se asigna al otro sub-grafo. El principal inconveniente de SB es el cálculo del vector de Fielder para grafos grandes. Algunos autores han reducido el tiempo de SB haciendo uso del paradigma multi-nivel. En particular, Barnard y Simon [BAS95] presentaron un método basado en bisección espectral recursiva (en inglés Recursive Spectral Bisection, RSB), que era capaz de acelerar SB un orden de magnitud sin perder calidad en términos de número de cortes.

El algoritmo de Farhat (en inglés Farhat's Greedy Algorithm, FGA) [FAH88] es un método rápido para computar bisecciones en grafos sin llevar a cabo un proceso recursivo. Como se puede observar en la Figura 3.5, su funcionamiento consiste en seleccionar un vértice aleatorio (v_{12}) y hacer crecer una región alrededor del mismo hasta que la mitad de vértices se incluyen en este sub-grafo, mientras que los restantes vértices se asignan a otro sub-grafo. La principal ventaja de FGA es su fácil generalización a cualquier número de sub-grafos.

El algoritmo de crecimiento de grafo avaro (en inglés Greedy Graph Growing Algorithm, GGGA) [KAR98a] es similar a FGA, pero GGGA añade vértices al sub-grafo de una forma determinista de acuerdo al beneficio obtenido por dicha adicción. Al igual que sucede en FGA, la selección aleatoria de un vértice inicial desde el cual iniciar el proceso de crecimiento suele tener un gran impacto en la calidad de la partición, razón por la cual es necesario llevar a cabo múltiples intentos o refinar la partición haciendo uso de métodos locales. Cabe indicar que tanto FGA como GGGA son difíciles de implementar en paralelo.

El paradigma multi-nivel (en inglés Multi-Level paradigm, ML) [HEN93]

es un esquema de optimización potente propuesto inicialmente para resolver el problema de repartición de grafos, aunque también ha sido satisfactoriamente adaptado a otros problemas de optimización [WAL04]. El uso de ML permite obtener particiones de alta calidad a costa de un sensible incremento en el tiempo de ejecución. Además, el paradigma multi-nivel no es fácilmente paralelizable. El funcionamiento general de de consta de tres fases diferenciadas, tal y como se muestra en la Figura 3.6.

i) Fase de agrupamiento. La primera fase consiste en reducir el tamaño del grafo original (G o G_0) preservando su estructura general. La creación de un grafo G_{i+1} (V_{i+1}, E_{i+1}) desde el grafo de un nivel superior, G_i (V_i , E_i) se lleva a cabo agrupando pares de vértices (también llamados nodos en el contexto multi-nivel) de G_i que están conectados por una arista, obteniendo un nuevo grafo G_{i+1}. Repitiendo este proceso, se van generando grafos con menor número de nodos, hasta que finalmente se obtiene un grafo agrupado, G_m, de unos pocos cientos o miles de nodos. Existen diversas técnicas de agrupamiento, entre las que destacamos tres: aleatoria, de aristas pesadas, y de clique pesado. La forma más sencilla de realizar este proceso es visitando vértices de forma aleatoria. El agrupamiento de aristas pesadas consiste en agrupar vértices cuya arista común tiene peso máximo con respecto a todas las aristas incidentes. La ventaja de esta alternativa radica en esconder las aristas más pesadas durante esta fase, es decir, el grafo resultante es construido mediante aristas con pesos reducidos. Por último, la técnica de agrupamiento de clique pesado se basa en agrupar vértices de forma que la densidad de la arista del multi-nodo creado es el mayor entre todos los multinodos que se pueden crear.

ii) Fase de repartición inicial. Una vez finaliza la fase de agrupamiento, es necesario realizar una repartición inicial del grafo agrupado, G_{NN}. Habitualmente, esta primera repartición se suele llevar a cabo mediante SB [POT90], FGA [FAH88], o GGGA [KAR98a]. Aunque la calidad de esta primera partición debe ser aceptable, no resulta tan determinate como al aplicar técnicas mono-nivel, ya que en el caso multi-nivel la estructura de vecindario se modifica conforme se asciende de nivel.

iii) Fase de refinamiento. La aplicación de un algoritmo de repartición como FGA o GGGA no es suficiente para obtener buenas particiones en grafos grandes. Por lo tanto, en dichos casos resulta necesario llevar a cabo una fase de refinamiento con el objetivo de mejorar la calidad de la partición inicial. Dicho proceso consiste en expandir el grafo agrupado, G_{NN}, de forma inversa a como fue agrupado hasta el grafo original G, a la vez que se aplica alguna técnica

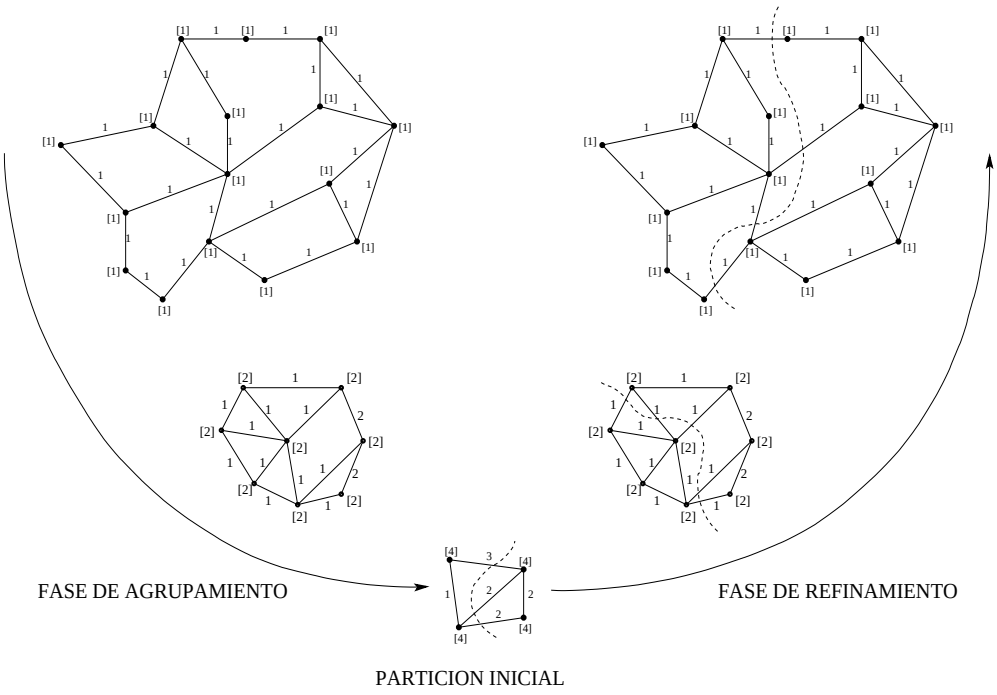

Figura 3.6: Paradigma multi-nivel.

de mejora. Los métodos clásicos suelen aplicar el algoritmo KL en cada fase de refinamiento. Sin embargo, recientemente algunos autores han planteado otras alternativas más sofisticadas como aplicar algoritmos evolutivos [SOP04], o enfriamiento simulado, búsqueda tabú, tal y como se describirá posteriormente.

3.1.2. Nuevas Tendencias en GPP

Repartición con Hiper-grafos

Como se comentó con anterioridad, los grafos pueden modelar una gran variedad de problemas. Sin embargo, algunos autores han extendido el modelo tradicional haciendo uso de hiper-grafos [BEE73], que son una generalización de grafos formados por vértices e hiper-aristas. Mientras que en el contexto de partición de grafos una arista conecta dos vértices, en hiper-grafos una hiper-arista es incidente con un número variable de vértices, es decir dos o más. El problema es particionar los vértices en SG sub-hiper-grafos equilibrados tal que el número de hiper-aristas conectando vértices de diferentes sub-hiper-grafos (cortes) es minimizado.

En la última década diferentes autores han mostrado que el modelo de hipergrafo ofrece una representación más adecuada que el modelo de grafos para determinados problemas. Por ejemplo, Catalyurek y Aykanat [CAT99] describieron las deficiencias del modelo tradicional de grafos para descomponer matrices dispersas para la multiplicación paralela matriz-vector, a la vez que demostraron que los hiper-grafos evitan dichas deficiencias. De forma similar, Karypis y col. [KAR99] han utilizado el paradigma multi-nivel en hiper-grafos obteniendo soluciones de alta calidad en aplicaciones VLSI. Algunos estudios en el contexto multi-objetivo también hacen uso de hiper-grafos [RUM02, SEL06].

Repartición de Grafos con Múltiples Restricciones

Recientes avances en simulación científica han destacado que la formulación tradicional del problema de repartición de grafos no es adecuada para asegurar su ejecución eficiente en algunas aplicaciones de alto rendimiento, como computaciones de múltiples fases [SCH00a]. Las simulaciones de múltiples fases consisten en una secuencia de fases computacionales, cada una de ellas separadas por una fase de sincronización. Debido a que la cantidad de computación en cada fase es diferente, el modelo tradicional de repartición de grafos no resulta efectivo. En su lugar, resulta necesario dividir el grafo de forma que los subgrafos en cada fase permanezcan equilibrados, mientras que el número de cortes entre diferentes sub-grafos en cada fase debe ser minimizado. Como indicaron Schloegel y col. [SCH00a], el objetivo es tratar de equilibrar las computaciones de cada fase como restricciones diferentes, de forma que el número de cortes sea minimizado sujeto a la restricción de que las computaciones llevadas a acabo en cada fase estén equilibradas. Algunas formulaciones con múltiples restricciones son descritas en detalle en [SCH00a, SCH02].

Repartición de Grafos con Múltiples Objetivos

Como comentamos con anterioridad, el principal inconveniente de la formulación mono-objetivo en repartición de grafos radica en la gran dependencia de la restricción de desequilibrio. Las Figuras 3.7(a) y 3.7(b) muestran dos particiones, s_1 y s_2, respectivamente. La representación de ambas en el espacio objetivo se muestra en la Figura 3.7(c). De esta forma, si el máximo desequilibrio permitido es DB_{max}=20 %, la solución s_2 es seleccionada como la mejor, mientras que si dicha restricción es DB_{max}=10 %, la mejor solución sería s_1, ya que s_2 tiene un desequilibrio prohibido a pesar de obtener menos cortes. Este factor se

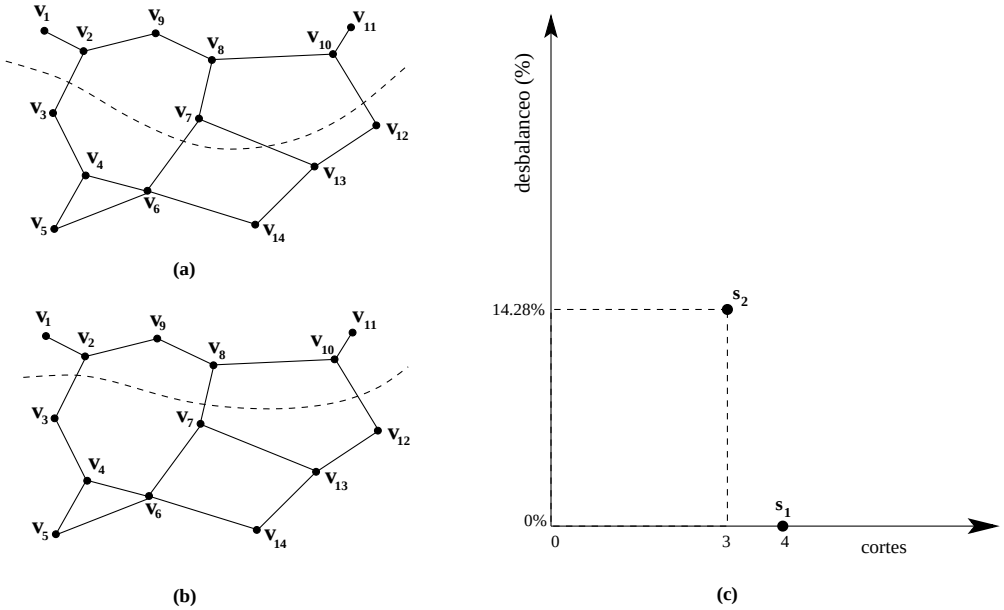

Figura 3.7: (a) Partición s1, (b) Partición s2, (c) espacio objetivo.

convierte en un handicap cuando se comparan varios métodos, ya que algunos pueden obtener muy buenos resultados para ciertos niveles de desequilibrio pero no para otros. Así, por ejemplo, en el archivo de repartición de grafos mantenido por Chris Walshaw [GPA] las mejores soluciones encontradas para varios grafos de prueba se clasifican de acuerdo a diferentes niveles de desequilibrio DB_{max}=$\{0\,\%,1\,\%,3\,\%,5\,\%\}$. Sin embargo, existen métodos que obtienen resultados para determinadas restricciones de desequilibrio pero no para otras.

Las formulaciones multi-objetivo del GPP han suscitado un gran interés entre los investigadores en los últimos años. Mientras que en optimización con un solo objetivo existe un óptimo global, en el caso multi-objetivo existen varias soluciones que optimizan simultáneamente todos los objetivos. Básicamente, mientras que en la formulación mono-objetivo el objetivo es minimizar el número de cortes, y el desquilibrio es considerado una restricción, en el contexto multi-objetivo ambos, cortes y desequilibrio se consideran objetivos a minimizar.

Selvakkumaran y Karypis [SEL06] describieron recientemente las dos principales aproximaciones para combinar múltiples objetivos en el GPP. La primera consiste en asignar diferentes prioridades a cada objetivo, de forma que siempre se prefiere aquella partición que optimice el objetivo de mayor prioridad, mientras que los objetivos de menor prioridad son utilizados en caso de igualdad

entre soluciones en el objetivo de máxima prioridad. La segunda aproximación consiste en agrupar los objetivos a optimizar en una función multi-objetivo. En [SEL06] se propone una función de coste similar a la descrita en la Ecuación 3.2. El primer término está asociado al número de cortes, mientras que el segundo término corresponde a la penalización asociada al grado de desequilibrio. El parámetro σ_k se utiliza para determinar la importancia relativa de los diferentes objetivos. Ababei y col. [ABE02] presentaron un algoritmo multi-objetivo para resolver un problema relacionado con circuitos. El principal inconveniente de estas aproximaciones radica en la gran dificultad de elegir valores adecuados para σ, ya que es dependiente del problema. Esta dificultad es aun mayor si consideramos el hecho de que ámbos objetivos tienen diferentes escalas.

$$coste(s) = \sigma_1 \cdot cortes(s) + \sigma_2 \cdot \%desquilibrio(s) \tag{3.5}$$

Con el objetivo de resolver las desventajas de esas formulaciones multi-objetivo basadas en pesos, en los últimos años algunos autores han utilizado el concepto de optimización de Pareto [GOL89] en el contexto de partición de grafos. Haciendo uso de optimización de Pareto, todos los objetivos (cortes y desequilibrio) tienen la misma importancia, por lo que se intentará encontrar un conjunto o frente de soluciones que optimicen ambos objetivos. Cabe indicar que hasta la actualidad se han realizado pocos estudios sobre formulaciones multi-objetivo del problema de repartición de grafos. Uno de los primeros trabajos que trataron la repartición de grafos en un contexto multi-objetivo haciendo uso del concepto de Pareto-optimalidad fue propuesto por Rummler y Apetrei [RUM02]. En concreto dichos autores adaptaron el algoritmo SPEA [ZIT99] haciendo uso de hiper-grafos. Los resultados obtenidos indicaron que MOEAs como SPEA obtienen resultados insatisfactorios debido principalmente a la redundancia en la representación de las soluciones para este problema, lo cual provoca que los operadores evolutivos no trabajen tan bien como en otros problemas.

Un aspecto importante a destacar a la hora de analizar el rendimiento de los algoritmos multi-objetivo haciendo uso de la optimización basada en frentes de Pareto radica en las métricas a utilizar a la hora de analizar la calidad de las soluciones. Tal y como se describió en el Capítulo 2, se van a aplicar conjuntamente las métricas de cobertura de conjuntos (SC) e hiper-volumen (H), ambas propuestas en [ZIT99]. A continuación vamos a describir su adaptación para el problema de repartición de grafos.

Definición 3.2. Cobertura de conjuntos (en inglés Set Coverage, SC). Esta métrica compara dos conjuntos de soluciones en términos de Pareto-dominancia relativa, es decir, presenta resultados del porcentaje de soluciones de un conjunto que son dominadas por alguna solución de otro conjunto. La Ecuación 3.3 describe el cálculo de esta métrica, considerando que s es una solución de S, s' una solución de S', y $|S'|$ el número de soluciones de S'. La función SC mapea dos conjuntos de soluciones (S,S') en el intervalo [0,1], de forma que el valor SC(S,S')=1 significa que todas las soluciones de S' son dominadas por una o más soluciones de S. En la Figura 3.8(a) se observa cómo S cubre a la mayor parte de las soluciones de S'.

$$SC(S, S') = \frac{|s' \in S', \quad \exists s \in S : \quad s \prec s'|}{|S'|} \tag{3.6}$$

Definición 3.3. Hiper-volumen (H). Sea S= (s_1, s_2, .., s_{tam}) un conjunto de soluciones no dominadas, la función $H(S)$ retorna el área del espacio objetivo (limitado por una solución de referencia, s_{ref}) que contiene soluciones que son dominadas por al menos uno de los miembros de S. Aunque conocida comúnmente como hiper-volumen, esta métrica fue propuesta por Zitzler y Thiele [ZIT99] con el nombre de tamaño del espacio cubierto (*size of the space covered* o *size of dominated space*). Posteriormente, Coello y col. [COE00] describieron esta métrica como la medida de Lebesgue ($\wedge$) [BUK05] aplicada a la unión de los hiper-cubos hc_i limitados por cada solución s_i y una solución de referencia s_{ref}, contabilizando el área común una sola vez. La Ecuación 3.4 describe la formulación matemática del Hiper-volumen:

$$H(S) = \Lambda(\{\bigcup_i (hc_i : s_i \in S)\}) = \Lambda(\bigcup_{s \in S} \{s' : s \prec s' \prec s_{ref}\}) \tag{3.7}$$

Como la formulación multi-objetivo del problema de repartición de grafos consiste en minimizar tanto el número de cortes como el desequilibrio, el punto de referencia es s_{ref}=(max_cortes,DB_{max}), es decir, el máximo número de cortes obtenido por cualquier solución del conjunto S, y el valor máximo de desequilibrio permitido, DB_{max}. Por lo tanto, la métrica H representa el hiper-volumen del área dominada por las soluciones de S, donde las áreas comunes son contabilizadas una sola vez. Las Figuras 3.8(b) y 3.8(c) muestran cómo el hiper-volumen es mayor en S que en S', por lo que se puede concluir, haciendo uso de esta métrica, que el frente S es de mayor calidad que S'.

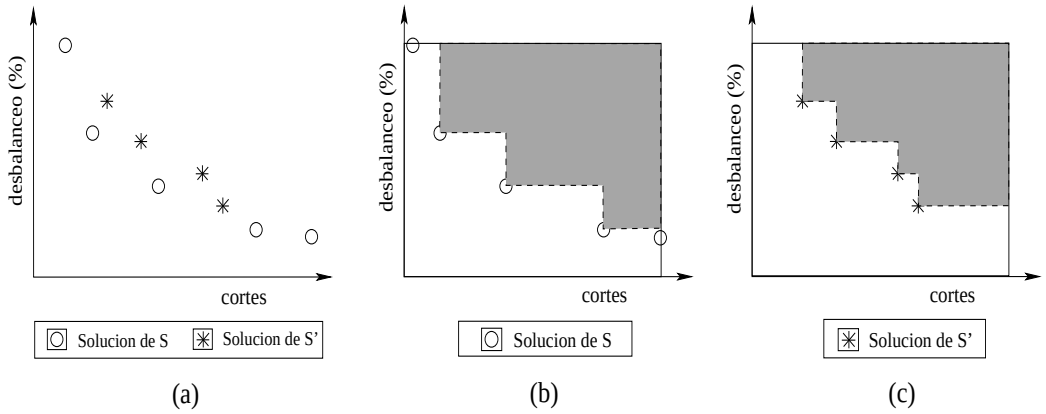

Figura 3.8: Métricas multi-objetivo utilizadas en GPP.

3.1.3. Grafos de Prueba Utilizados

La Tabla A.1 ofrece una breve descripción de los grafos utilizados en los análisis experimentales. En concreto se muestra su número de vértices, número de aristas, máxima conectividad (max) (número de vecinos del vértice con el mayor vecindario), mínima conectividad (min), conectividad media (avg), y tamaño del archivo. Dichos grafos, junto sus descripciones, y las mejores soluciones encontradas para ellos clasificados por niveles de desequilibrio máximo (DB_{max}=$\{0\,\%,1\,\%,3\,\%,5\,\%\}$), se encuentran en [GPA].

3.2. Problema del Diseño de Redes Malladas de Distribución de Agua (WDND)

El diseño óptimo de redes de distribución de agua consiste en encontrar la forma más eficiente de suministrar agua a los usuarios cumpliendo ciertos requisitos, lo cual implica obtener configuraciones compromiso entre eficiencia y coste. En el diseño de este tipo de redes se han de tomar un gran número de decisiones, como son su localización física, el tamaño de sus componentes (diámetros de tuberías, capacidad de los tanques y potencia de las bombas), topología y conectividad, etc.

Las redes de distribución de agua han sido diseñadas tradicionalmente mediante estructuras ramificadas, debido a su facilidad de diseño y reducido coste. La optimización de los sistemas de redes malladas de distribución de agua es mucho más compleja que la optimización de redes ramificadas, ya que los flujos de las tuberías son variables no conocidas a priori. Por otro lado, considerar la fiabilidad de la red suele conllevar un incremento en el coste total de diseño. Sin embargo, diversos sistemas de distribución de agua, como por ejemplo cultivos hortofrutícolas en invernaderos, requieren su consideración debido al uso de sistemas de riego de gran frecuencia. Por esta razón, las redes malladas de distribución de agua parecen ser una alternativa más adecuada que las ramificadas debido a que compensan el sensible incremento en el coste de diseño con un incremento en la fiabilidad de las mismas [SAV97].

Este problema entra dentro de la clase de problemas NP-duros [GUP93]. Es un problema no lineal, con restricciones y multi-modal. Como resultado del análisis llevado a cabo por diferentes autores en las últimas décadas, se han propuesto un gran número de técnicas. Además de aplicar ténicas de optimización global [HOS95] como por ejemplo [SHR98], o técnicas no lineales [VAR97], también se han utilizado una gran variedad de técnicas heurísticas para resolver este problema [MON99, CUS99, VAI00, GEE01, MAI03, REC06].

La mayoría de las técnicas propuestas para resolver este problema utilizan una formulación mono-objetivo con restricciones. En concreto, lo habitual es minimizar el coste de diseño, siempre que dicha red cumpla unas determinadas restricciones, referidas normalmente a presiones mínimas en los nodos de demanda. De forma habitual, el problema del diseño óptimo de redes de distribución de agua se considera como un problema de optimización donde las variables de decisión son los diámetros de las tuberías, mientras que la localización física de

la red, conectividad y demandas en los nodos son impuestas por el problema [SAV97].

Sin embargo, existen otras formulaciones más recientes [FAR03, FAR05] en las cuales se consideran simultáneamente, no sólo la minimización del coste de diseño de la red, sino también la maximización de la fiabilidad de dicho diseño. El gran inconveniente de esta formulación multi-objetivo radica en la existencia de múltiples criterios para determinar dicha eficiencia, como lo demuestra el hecho de que se han propuesto diferentes índices de de fiabilidad o elasticidad [MAY00], sin que hasta el momento exista ninguna que haya sido aceptada de forma universal.

3.2.1. Descripción del Problema

Como comentamos con anterioridad, en la formulación tradicional de este problema [SAV97, MON99] las variables de decisión son los diámetros de las tuberías, y el objetivo es minimizar el coste de la red. La localización física de la red, conectividad y demandas son restricciones impuestas por el problema, por lo que no serán consideradas en los algoritmos de optimización implementados. Además, el problema está restringido por las leyes físicas de masa y energía. Por otro lado, se requiere una presión mínima en los nodos de demanda, que en nuestra implementación será de 30 metros de caudal. Con el objetivo de incluir esta restricción se hace uso de una constante de penalización (K_p) de forma que aquellos diseños en los cuales los nodos no alcancen la presión mínima requerida serán penalizados. La Ecuación 3.5 ofrece la descripción matemática para calcular el coste total de la red.

$$f = \sum_{i=1}^{n_d} c_i L_i + K_p \sum_{j=1}^{n_n} max[(hr_j - h_j), 0] \tag{3.8}$$

donde: f es la función objetivo a minimizar, n_d es el número de diámetros de tubería disponibles, c_i es el coste por unidad de longitud de las tuberías de diámetro i, L_i es la longitud total de las tuberías de diámetro i, K_p es la constante de penalización, n_n es el número de nodos, hr_j es la presión de cabecera requerida en el nodo j, y h_j es la presión de cabecera actual computada por el simulador hidráulico para el nodo j. En los experimentos se ha utilizado una constante de penalización muy elevada (K_p=100000) con la idea de descartar aquellas soluciones con presiones por debajo de los requisitos exigidos (30 metros de caudal).

3.2.2. Nuevas Tendencias en WDND

La formulación mono-objetivo con restricciones que acabamos de describir ha sido la más utilizada hasta la fecha. No obstante, dicha formulación presenta algunos inconvenientes. El principal, al igual que en el problema de partición de grafos, radica en la gran alta dependencia de las restricciones de demanda en los nodos. En concreto, la elección del parámetro K_p puede tener efectos indeseados, como por ejemplo que haya diseños poco costosos económicamente que sean descartados directamente debido a que no se alcanza la presión mínima en uno o pocos nodos. Una forma de evitar este inconveniente consiste en eliminar dicha restricción y plantearlo como otro objetivo a optimizar. Esto se realiza mediante el uso de un parámetro denominado fiabilidad (*resilience*). Así pues, la formulación multi-objetivo trata de minimizar el coste de diseño y maximizar de la fiabilidad del sistema.

En referencia al coste de la red, esta se define de forma similar al caso mono-objetivo, es decir, como la suma del coste del total de tuberías que se han utilizado para diseñar la red, considerando la longitud y el grosor de las mismas. La Ecuación 3.6 muestra cómo se calcula este objetivo.

$$I_c = \sum_{i=1}^{n_d} c_i L_i \tag{3.9}$$

donde: I_c es el coste económico de la red, n_d es el número de posibles diámetros de tuberías disponibles, c_i es el coste, por unidad de longitud, de las tuberías de diámetro i en la red, L_i es la longitud total de las tuberías de diámetro i en la red.

En cuanto a la fiabilidad, hemos utilizado el índice propuesto por Todini [TOD00], y posteriormente utilizado por otros autores [FAR05]. La Ecuación 3.7 ofrece una descripción de dicho índice.

$$I_r = \frac{\sum_{j=1}^{n_n} q_j(h_j - hr_j)}{(\sum_{tq=1}^{n_r} Q_{tq}H_{tq} + \sum_{b=1}^{n_b} \frac{PW_b}{\gamma}) - \sum_{j=1}^{n_n} q_j hr_j} \tag{3.10}$$

donde: I_r es el indice de fiabilidad, n_n el número de nodos, q_j el flujo en el nodo j, h_j la presión disponible en el nodo j, hr_j la presión requerida en el nodo j, n_r el número de reservas (tanques), n_b el número de bombas, Q_{tq} el flujo que sale del tq-ésimo tanque cuando está suministrando agua al sistema, H_{tq} la

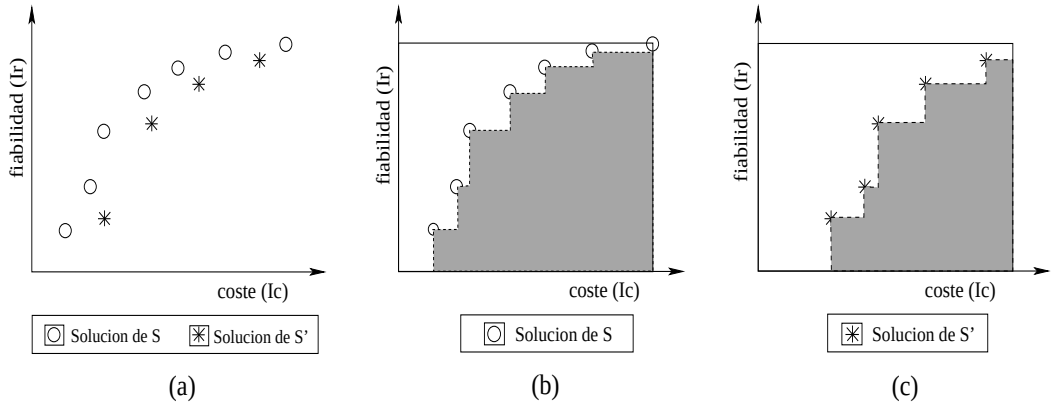

Figura 3.9: Métricas multi-objetivo utilizadas en WDND.

elevación del nivel de agua en el tq-ésimo tanque, PW_b la potencia introducida por las bombas en el sistema, y γ el peso específico del agua.

De forma similar al problema de repartición de grafos vamos a utilizar las métricas de cobertura de conjuntos (SC) y de hiper-volumen (H). Sin embargo, mientras que en el problema de repartición de grafos tanto el coste como el desequilibrio eran objetivos a minimizar, en el problema del diseño de redes de distribución de agua hay que minimizar el coste y maximizar la fiabilidad. Esto no implica ningun cambio en lo referente a la métrica de cobertura de conjuntos (SC), siendo válida la formulación descrita en la Definición 3.2. Sin embargo, el cálculo del hiper-volumen difiere respecto al problema de repartición de grafos, ya que el punto de referencia (s_{ref}) en este caso viene determinado por s_{ref}=(max_coste,0), es decir, el máximo coste de diseño obtenido por cualquier solución del conjunto S, y el mínimo valor de fiabilidad admitido, que en estas formulaciones se considera I_r=0.

Las Figuras 3.9(b) y 3.9(c) muestran como el hiper-volumen es mayor en S que en S', lo cual indica que, haciendo uso de esta métrica, S es considerado de más calidad que S'.

3.2.3. Redes de Distribución de Agua Utilizadas

Con el objetivo de evaluar el rendimiento de las técnicas meta-heurísticas implementadas en este problema se hace uso de tres redes malladas de distribución de agua, cada una de las cuales tiene características particulares como la topología, tamaño, o utilidad. Para comparar en la medida de lo posible con

resultados obtenidos por otros autores, se utilizan los parámetros que incluye por defecto el simulador EPANET 2.00.07 ([ROS00]).

La pérdida de carga en una tubería debida a la fricción por el paso del agua, puede calcularse en EPANET haciendo uso de tres fórmulas diferentes: Hazen-Williams, Darcy-Weisbach, y Chezy-Manning [ROS00, MOT05]. A continuación describimos las dos primeras, que son las utilizadas a la hora de evaluar las redes de prueba del problema.

La ecuación de Hazen-Williams [ROS00, MOT05] viene descrita en la Ecuación 3.8., donde C es el coeficiente de rugosidad de Hazen-Williams, d_i es el diámetro de la tubería i, L_i la longitud de la tubiería i. Los parámetros por defecto incluidos por EPANET y también utilizados en nuestro estudio experimental son: α=4.727, a=1.852, b=4.871. Además, para todos los experimentos vamos a utilizar el coeficiente de rugosidad es C=130.

$$HW = \alpha C^{-a} d_i^{-b} L_i \tag{3.11}$$

La ecuación de Darcy-Weisbach [ROS00, MOT05], viene descrita en la Ecuación 3.9., donde f_f es el factor de fricción (dependiente de ε, d_i, y q), ε es el coeficiente de rugosidad de Darcy-Weisbach, d_i es el diámetro de la tubería i, qq_i es el caudal de la tubería i, y L_i la longitud de la tubería i. Los parámetros utilizados en nuestro estudio experimental son: β=0.0252, ε=0.85. La base de datos de tuberías disponibles está compuesta por 10 tuberías de PVC comerciales con diámetros nominales de entre 125 y 600 mm.

$$DW = \beta f_f(\varepsilon, d_i, qq_i) d_i^{-5} L_i \tag{3.12}$$

Red de Alperovits-Shamir

La primera de las tres redes que se utiliza en los experimentos corresponde a la red de Alperovits-Shamir [ALP77]. Esta es una red mallada simple de 2 lazos, 7 nodos y 8 tuberías. La red obtiene el agua de una sola presa con una elevación de 210 metros. Para el diseño de esta red se pueden escoger entre 14 diámetros comerciales, por lo que existen $14^8 = 1,4758 * 10^9$ configuraciones posibles. La demanda mínima en cada nodo para la implementación mono-objetivo es de 30 metros de caudal, mientras que otros factores como el coste por unidad de longitud, han sido definidas con todo detalle en trabajos previos [ALP77]. La descripción física de la red se muestra en la Figura 3.10.

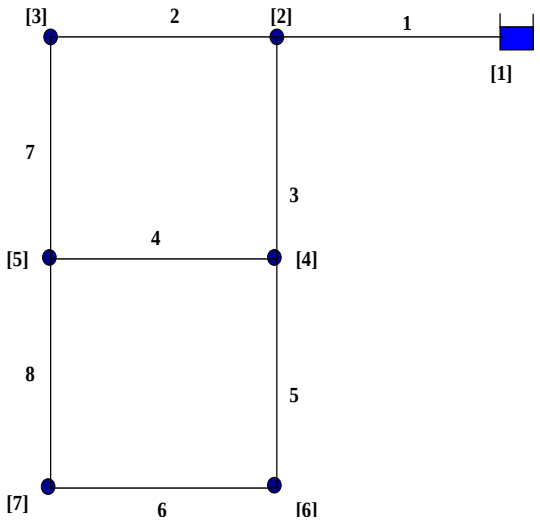

Figura 3.10: Red de Alperovits-Shamir.

Red de Hanoi

Otra de las redes de prueba utilizadas es la red de tuberías para el suministro de agua de la ciudad de Hanoi (Vietnam), propuesta por Fujiwara y Khang [FUJ90]. Esta red, de tamaño intermedio, consta de 3 lazos, 32 nodos, y 34 tuberías. No se consideran estaciones de bombeo ya que hay una sola fuente fija con una elevación de 100 metros. En este caso, la presión mínima requerida en los nodos se establece también en 30 metros de caudal. Se pueden escoger entre 6 diámetros comerciales, por lo que el total de configuraciones posibles es $6^{34} = 2,8651\ 10^{26}$. En esta red la ecuación de Hazen-Williams también se utilizan los parámetros por defecto incluidos en EPANET, y además el coeficiente de rugosidad C=130. La descripción física de la red se muestra en la Figura 3.11.

Red de Balerma

La tercera y última red de prueba que hemos utilizado en el estudio experimental corresponde a una red de riego de gran tamaño. Esta red, denominada red de Balerma, suministra agua al distrito de riego Sol-poniente, en el municipio de Balerma, provincia de Almería. Es una red multi-fuente (con 4 presas), que contiene 443 nodos (bocas de riego) cuya agua proviene de 4 reservas. Tiene un total de 454 tuberías organizadas en 8 lazos, tal y como se muestra en la Figura 3.12. Por lo tanto, el total de configuraciones posibles para esta red es 10^{454},

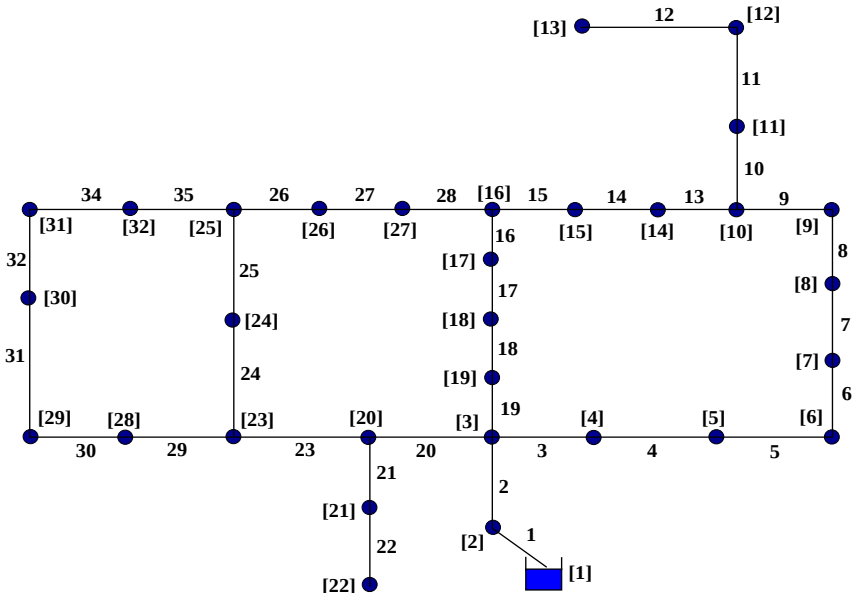

Figura 3.11: Red de Hanoi.

por lo que nos podemos hacer una clara idea de la extrema complejidad de su optimización. La presión mínima requerida es de 20 metros de caudal. Podemos encontrar más detalles sobre esta red en el trabajo de Reca y Martínez [REC06]. Además, la descripción de la red, así como la base de datos de tuberías pueden ser descargadas desde el apartado de materiales auxiliares de la Unión Geofísica Americana (AGU) [AGU].

Figura 3.12: Red de Balerma.

3.3. Conclusiones

Un grafo es una generalización de un gran número de problemas reales que consiste en un conjunto de vértices que están interconectados entre sí mediante aristas. La formulación clásica del problema de repartición de grafos consiste en dividir un grafo en un número arbitrario de sub-grafos tal que se minimice el número de aristas que unen vértices pertenecientes a diferentes sub-grafos, a la vez que la suma de los pesos de los vértices en cada sub-grafo permanece equilibrado. Existe un elevado número de técnicas para llevar a cabo la repartición. Algunas de ellas hacen uso de información geométrica del grafo, mientras que otros hacen uso de la conectividad del mismo. Los métodos geométricos son muy rápidos, aunque al no hacer uso de la información sobre la conectividad suelen obtener particiones con un número cortes superior. La tendencia de los últimos años es la de utilizar el paradigma multi-nivel en combinación con otros métodos de refinamiento.

El diseño de redes de distribución de agua consiste básicamente en encontrar la forma más eficiente de suministrar agua a los usuarios cumpliendo una serie de requisitos. El enfoque tradicional de este problema corresponde a un problema mono-objetivo con restricciones, donde el objetivo es reducir el coste económico del diseño a la vez que se cumplen una serie de restricciones, normalmente referidas a presiones mínimas en los nodos de demanda. Debido a la complejidad del problema, el uso de técnicas heurísticas está plenamente justificado. Sin embargo, en este modelo resulta de gran dificultad establecer un compromiso adecuado entre el coste económico del diseño y su eficiencia técnica. Por esta razón, recientemente algunos autores han extendido este enfoque tradicional a un modelo multi-objetivo, con lo que además de minimizar el coste económico de la red se intenta maximizar la calidad de la misma haciendo uso de ciertos índices que evaluan la fiabilidad del sistema.

La elección de estos problema para evaluar el comportamiento de los algoritmos propuestos en esta tesis ha venido determinada por tres razones principales. La primera es que se trata de problemas combinatorios para los cuales, pese a haber sido tratados extensivamente en la literatura, no existen métodos exactos que permitan resolverlo de forma totalmente eficiente, dadas sus complejidades computacionales (el problema de repartición de grafos es NP-completo [GAR79], y el problema de redes de distribución de agua es NP-duro [GUP93]). Por otro lado, cabe indicar que hasta la fecha se han realizado relativamente pocos estudios de ambos problemas haciendo uso de formulaciones multi-objetivo basadas

en optimización de Pareto. Finalmente, ambos problemas resultan de especial interés dada su gran aplicabilidad práctica. Por un lado el problema de repartición de grafos se puede extender a múltiples aplicaciones reales, como repartición de circuitos, repartición de mallas, etc., mientras que el diseño de redes de distribución de agua abarca una gran cantidad de sistemas, como suministro urbano, redes de riego, etc.

Una vez descritas las formulaciones mono-objetivo y multi-objetivo de ambos, así como de las diversas métricas de rendimiento, e instancias de prueba utilizadas, resultará inmediato el análisis de resultados de los procedimientos presentados en capítulos posteriores.

Capítulo 4

Algoritmos Mono-objetivo: Descripción y Resultados

En capítulos anteriores se introdujeron los conceptos de optimización necesarios a la hora de abordar cualquier problema de optimización, tanto mono-objetivo como multi-objetivo. Además, se han descrito con detalle los dos problemas tratados en esta tesis y sobre los cuales se han evaluado los diferentes procedimientos heurísticos presentados en los siguientes capítulos. Con esos ingredientes, ya nos encontramos en disposición de presentar formalmente las características de dichos procedimientos y analizar los resultados obtenidos por los mismos en los problemas de optimización descritos anteriormente. En concreto, el presente capítulo describe las técnicas heurísticas mono-objetivo propuestas.

4.1. rMSATS: refined Mixed Simulated Annealing and Tabu Search

En esta sección se presenta rMSATS [GIL02], que generaliza la heurística para repartición de circuitos MSATS [GIL96], para tratar el problema de repartición de grafos en su versión mono-objetivo. El procedimiento rMSATS combina aspectos de SA y TS, y su funcionamiento se puede describir en dos fases, tal y como se describe a continuación.

4.1.1. Descripción del Procedimiento

i) Fase de reparticionamiento inicial. En una primera fase de reparticionamiento inicial rMSATS utiliza un algoritmo de crecimiento basado en FGA [FAH88], descrito en el Capítulo 3. Como ya se comentó anteriormente, la posición del vértice inicial en FGA determina la estructura de la partición inicial, por lo que se puede intentar realizar la repartición desde diferentes vértices iniciales con el objetivo de incrementar la diversidad en el proceso de búsqueda.

ii) Fase de refinamiento. La aplicación de un algoritmo como FGA no es suficiente para obtener una partición de calidad. Por lo tanto, es necesario llevar a cabo una fase de refinamiento o mejora en la cual se explore de forma eficiente el espacio de búsqueda. Algunas técnicas, como KL [KER70] o FM [FID82] tienen como inconveniente que la búsqueda puede quedar atrapado en óptimos locales. Con el objetivo de solucionar este problema, rMSATS utiliza enfriamiento simulado (SA) [KIR83] y búsqueda tabú (TS) [GLO93]. El uso combinado de ambas da como resultado una estrategia híbrida que permite al proceso de búsqueda escapar de óptimos locales, evitando al mismo tiempo la aparición de ciclos. El funcionamiento de rMSATS consiste en utilizar SA conforme se vayan moviendo vértices limítrofes hacia sub-grafos vecinos de forma que se acepten utilizando el criterio de Metrópolis [MET53]. Este criterio tiene dos parámetros de entrada: la diferencia en número de cortes provocado por dicho movimiento, y el valor actual de temperatura (t). Al inicio del proceso (temperaturas altas) se permitirá escapar de óptimos locales mediante la aceptación de movimientos de vértices entre sub-grafos limítrofes que incrementen el número de cortes, mientras que al final del proceso (temperaturas bajas) sólo se aceptarán movimientos de vértices que disminuyan el número de cortes, todo ello respetando las restricciones de desequilibrio. Por otro lado, el uso de un vecindario reducido para explorar el espacio de búsqueda puede favorecer la aparición de ciclos.

Con el objetivo de evitar este problema, se aplica TS como complemento a SA. Así, cuando la función de Metrópolis acepta un movimiento que empeora la función de coste, el vértice movido se incluye en la lista tabú (LT). El conjunto de vértices incluidos en LT no se pueden mover en la iteración actual, aunque se eliminan de esta lista en la siguiente iteración (memoria a corto plazo con ψ=1). Los resultados experimentales indican que el uso conjunto de ambas técnicas mejora los resultados obtenidos cuando se aplica solamente SA o TS.

Algoritmo 4.1.1

Begin rMSATS (G, SG, DB_{max}, T_i, T_{enfr}, T_{parada})
  s←Obtener una partición inicial de G mediante FGA;
  t←T_i; LT←ϕ;
  cortes_mejor_partición←∞;
  Hacer
    Para cada (vértice limítrofe v) **hacer**
      aux←reducción de cortes al mover v;
      Si ((Metropolis(aux,t)=TRUE)
        & (nuevo_desequilibrio≤DB_{max})) **entonces**
        Mover_vértice (v, sub-grafo limítrofe);
        Si (aux≤0) **entonces**
          LT←$LT \cup v$;
        Si (cortes(s)≤cortes_mejor_partición) **entonces**
          mejor_partición←s;
          cortes_mejor_partición←cortes(s);
        t←$t \cdot T_{enfr}$;
  Mientras ($t \geq T_{parada}$);
  Retornar s;
End rMSATS

El Algoritmo 4.1.1 muestra el funcionamiento del procedimiento rMSATS. Una vez obtenidos los parámetros de entrada, la partición inicial se obtiene haciendo uso del algoritmo FGA [FAH88]. Mientras la condición de parada no se cumpla, es decir, la temperatura (t) sea mayor que el umbral establecido (T_{parada}), se itera de la siguiente forma. En cada iteración de este bucle los

vértices limítrofes son evaluados de forma que se aceptan o rechazan aplicando el criterio de Metrópolis, el cual a su vez hace uso de t y de la reducción/incremento en el número de cortes. Si el movimiento se acepta, se testea si dicho movimiento implica un incremento en el número de cortes, en cuyo caso el vértice es añadido a LT. Previamente a comenzar la siguiente iteración, la temperatura es actualizada haciendo uso del coeficiente de enfriamiento T_{enfr}. Finalmente, se retorna la mejor solución encontrada.

4.2. MLrMSATS: Multi-level rMSATS

Como se describió en el Capítulo 3, el paradigma multinivel [HEN93, KAR98a] se ha convertido en una estrategia muy eficiente en la resolución del problema de repartición de grafos. A continuación describimos el funcionamiento de MLrMSATS [BAÑ03], la versión multinivel de rMSATS, que consta de tres fases diferenciadas, tal y como describimos a continuación.

4.2.1. Descripción del Procedimiento

i) Fase de agrupamiento del grafo. La creación de un grafo agrupado G_{i+1} (V_{i+1}, E_{i+1}) desde el grafo de nivel previo G_i (V_i , E_i), consiste en agrupar pares de nodos de G_i que están interconectados por una arista. Repitiendo este proceso se van generando grafos con menos nodos hasta que se obtiene el grafo más agrupado. En MLrMSATS, la fase de agrupamiento finaliza cuando el número de nodos cae por debajo de un umbral dado $\mu \cdot SG$, donde μ es un parámetro de entrada. Al igual que en trabajos previos [KAR98b], en nuestros experimentos hemos utilizado el valor μ=15.

MLrMSATS utiliza dos posibles estrategias de agrupamiento de nodos. La primera, denominada agrupación aleatoria (en inglés random matching, RM) [KAR98a] selecciona un nodo de forma aleatoria, que es agrupado con otro nodo adyacente también seleccionado aleatoriamente. La segunda estrategia, denominada agrupación pesada (en inglés HEavy Matching, HEM) [KAR98a] consiste en agrupar los vértices en función del peso de las aristas que los unen de forma que se toma un vértice no agrupado en el nivel actual y se une con aquél vértice vecino no agrupado cuya arista de unión entre ambos tenga un peso más elevado. De esta forma se ocultan las aristas más pesadas durante el agrupamiento, con lo que el grafo resultante estará formado por aristas con peso inferior, y por tanto al realizar la repartición el número de cortes tiende a reducirse. Indicar

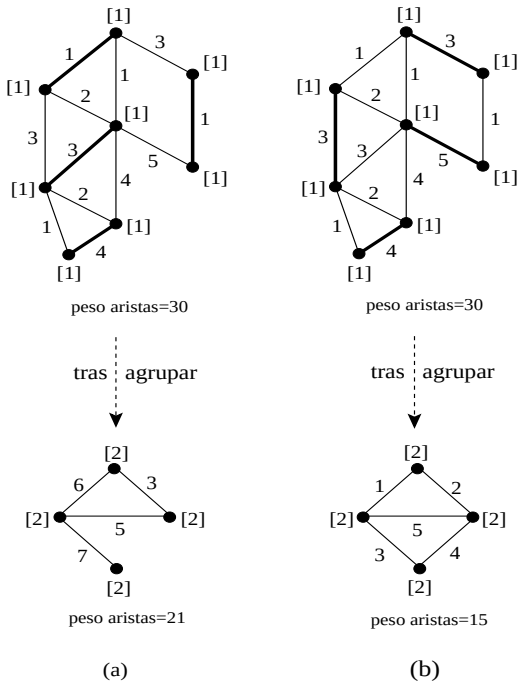

Figura 4.1: (a) Random y (b) HEavy Matching.

que la potencialidad de HEM sólo se puede aprovechar al realizar el agrupamiento en más de dos niveles, ya que en el nivel superior los vértices y aristas utilizados tienen peso unidad. Un ejemplo acerca del comportamiento de ambas alternativas de agrupamiento lo podemos ver en la Figura 4.1. Como se observa, en este ejemplo la fase de agrupamiento utilizando HEM ayuda a ocultar las aristas pesadas razón por la cual el grafo agrupado tiene un peso total de aristas igual a 15, mientras que en RM es 21.

En ambas alternativas puede ocurrir que al visitar los nodos para ser agrupados estos no tengan nodos vecinos no agrupados en este nivel con los que unirse, por lo que pasarán directamente como nodos del nivel inferior, lo cual provoca un desequilibrio en el peso de los nodos. Este proceso recursivo puede provocar que el desequilibrio se incremente en niveles inferiores. Así, considerando que el grafo de engrada (G) está formado por vértices con peso unitario en el nivel i los pesos de los nodos del grafo G_i puedan oscilar en el intervalo $[1, 2^{i-1}]$. Para atenuar este inconveniente y conseguir la homogenización de los pesos de los nodos de niveles inferiores, MLrMSATS visita primero los nodos con menor peso, es decir, aquellos que pudieron quedar aislados en niveles previos.

ii) Fase de reparticionamiento inicial. Al igual que en el procedimiento básico rMSATS, el grafo agrupado va a ser dividido haciendo uso de FGA [FAH88]. Debido a que el grafo agrupado tiene un número de vértices muy inferior al grafo original, el algoritmo FGA se ejecutará en un tiempo reducido. Es precisamente en esta fase donde tiene efecto el hecho de visitar los nodos en orden inverso de peso, ya que se supone que el grafo agrupado obtenido tendrá nodos más equilibrados, con la ventaja que esto conlleva.

iii) Fase de refinamiento. Una vez agrupado el grafo original, la última fase consiste en una vuelta atrás desagrupando los grafos agrupados y aplicando un procedimiento de búsqueda que permita refinar la calidad de dichos niveles intermedios hasta llegar al grafo original. Es aquí donde entra en juego el aspecto más novedoso de MLrMSATS. Mientras la mayoría de las técnicas multi-nivel aplican métodos de búsqueda local simples [KER70, FID82], MLrMSATS aplica rMSATS en cada uno de los niveles agrupados, de forma que en cada nivel se optimiza la solución obtenida en el nivel previo haciendo uso de los valores de enfriamiento originales. Este proceso se repite hasta optimizar el grafo de nivel más elevado, obteniendo de esta forma la solución final.

Algoritmo 4.2.1

Begin MLrMSATS (G, SG, DB_{max}, T_i, T_{enfr}, T_{parada}, *estr_agrup*, μ)
  $NN \leftarrow$ `Calcular_Número_Niveles_Agrupamiento(`SG`,`μ`)`
  `Agrupar_Grafo(`G`,`NN`,estr_agrup);`
  $s \leftarrow$ `Obtener partición inicial de` G`[`NN`] usando FGA;`
  Desde `Nivel_Actual=`NN **hasta** `Nivel_Actual=1`
    `Aplicar rMSATS(`G`[`NN`],`SG`,`DB_{max}`,`T_i`,`T_{enfr}`);`
    `Desagrupar` G `un nivel;`
  `Retornar` s`;`
End MLrMSATS

El Algoritmo 4.2.1 describe el pseudo-código de MLrMSATS. Los parámetros de entrada son el grafo a dividir, el número de sub-grafos a dividir, el desequilibrio máximo, la temperatura inicial y el factor de enfriamiento, así como la estrategia de agrupamiento y el parámetro μ. En primer lugar, se calcula el número de niveles de agrupamiento en función de los parámetros μ y SG, tal y como explicamos anteriormente. Posteriormente, se aplica la fase de agrupamiento utilizando HEM o RM. Una vez realizada la agrupación, se aplica FGA

en el grafo de nivel inferior. Es en este momento en el cual comienza la fase de refinamiento haciendo uso de rMSATS, de forma que cada vez que se asciende de nivel se re-inicializan los valores originales de T_i y T_{enfr}. Tras refinar el nivel superior, se devuelve la mejor solución obtenida.

4.2.2. Análisis Experimental: rMSATS y MLrMSATS

Todos los algoritmos implementados en la resolución del problema de repartición de grafos han sido desarrollados haciendo uso del lenguaje de programación C estándar bajo el sistema operativo Linux y ejecutadas en estaciones de trabajo, que en función del instante en cuestión han oscilado desde Pentium II a 333 MHz hasta Pentium IV a 2.4 GHz.

La Tablas A.2, A.3, A.4, y A.5 muestran los resultados obtenidos por rMSATS frente a FGA [FAH88], y METIS [MES] que es un software de libre distribución para particionar grafos. En concreto, hemos comparado rMSATS con dos procedimientos: *pMetis* (basada en la bisección recursiva multi-nivel) y *kMetis* (basada en el repartición multi-nivel de *SG*-vias) [KAR98c]. Se han utilizado los grafos de prueba descritos en la Tabla A.1. Para todas las ejecuciones en el problema de repartición de grafos mono-objetivo se va a considerar un desequilibrio máximo DB_{max}=5 %, o lo que es lo mismo, el peso del grafo más pesado esta limitado por el valor $M \leq ((W_V/SG)^*(1.05))$. En estos experimentos hemos utilizado valores de temperatura iniciales en el intervalo T_i=[50,150], pero los resultados obtenidos han mostrado que este factor tiene poca incidencia sobre la calidad de las particiones obtenidas. Los resultados obtenidos muestran como rMSATS mejora claramente a FGA [FAH88] en todos los casos. Por otro lado, al comparar los resultados de rMSATS frente a *kMetis* y *pMetis* [MES], se observa como, en la amplia mayoría de casos, rMSATS obtiene el mejor resultado en términos de número de cortes. Debemos considerar el hecho de que los resultados de *kMetis* y *pMetis* se han obtenido tras ejecutar ambos programas haciendo uso de las configuraciones por defecto. Esto provoca que no se pueda elegir un grado de desequilibrio máximo, razón por la cual no es posible conocer los resultados de todos los métodos comparados haciendo uso del mismo nivel de desequilibrio.

En cuanto al rendimiento de MLrMSATS podemos concluir que, en la mayoría de los casos, la calidad de las particiones en referencia al número de cortes viene determinada por los parámetros de enfriamiento seleccionados, tal y como muestra la Figura 4.2. En dicha gráfica se puede observar como el uso de un

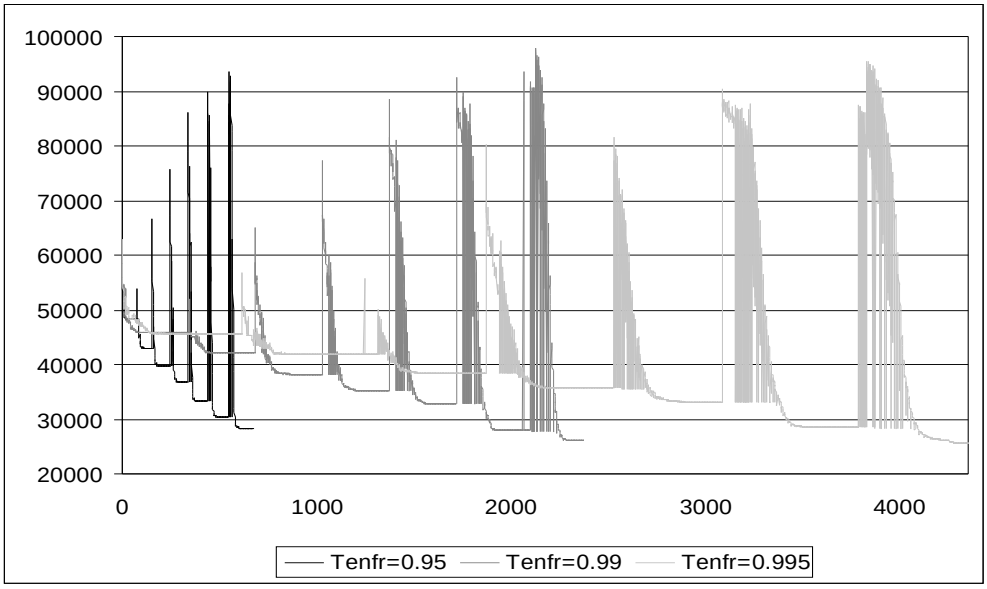

Figura 4.2: Efecto de modificar Tenfr en MLrMSATS.

esquema de enfriamiento lento haciendo uso de factores de enfriamiento próximos a la unidad (0.995) permite obtener mejores soluciones que haciendo uso de factores de enfriamiento inferiores (0.99 ó 0.95) a costa de un incremento del número de iteraciones, y por tanto del tiempo de ejecución. Sin embargo, dicho incremento en el tiempo de ejecución está plenamente justificado para determinadas aplicaciones reales, principalmente aquellas de diseño (*layout*). Debemos indicar que la principal ventaja del esquema multi-nivel es que este trabaja indirectamente como un procedimiento de vecindario variable, ya que en cada nivel de refinamiento rMSATS se aplica sobre un grafo con diferente estructura. Esto permite que, tal y como se muestra en la Figura 4.2, un determinado movimiento que permita escapar de un óptimo local puede no ser aceptado en un cierto nivel al superar el desequilibrio máximo, aunque si en un nivel superior.

La Figura 4.3 muestra un análisis del comportamiento de MLrMSATS utilizando las dos estrategias de agrupamiento descritas con anterioridad. En ambos casos, la temperatura inicial en la fase de refinamiento es T_i=100. En dicha figura, el eje de ordenadas representa el incremento en el número de cortes considerando que el valor 1 de dicho eje representa el número de cortes de la mejor solución obtenida, la cual actuará como referencia para calcular el incremento del número de cortes obtenido en el resto de configuraciones para dicho grafo

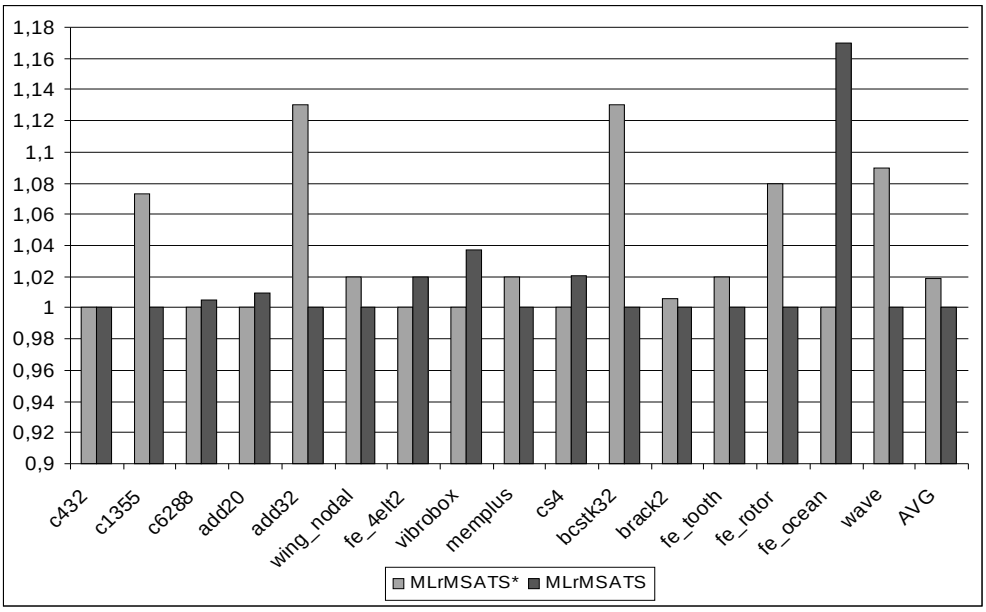

Figura 4.3: Comparativa entre RM (MLrMSATS*) y HEM (MLrMSATS).

(un valor de 1,04 en dicho eje representa un incremento del 4 % en el número de cortes). Tomando en cuenta dichas consideraciones, los resultados mostrados en la Figura 4.3 muestran como la estrategia HEM (MLrMSATS) mejora los resultados obtenidos por RM (MLrMSATS*) en la mayoría de los casos. En media (AVG), HEM reduce en casi un 2 % el número de cortes obtenido por RM.

Otro análisis de interés consiste en evaluar la mejora obtenida, en términos de número de cortes, de aplicar el esquema multinivel (MLrMSATS) frente a su aplicación en el grafo original (rMSATS). La Figura 4.4 muestra la comparativa entre ambos métodos tomando la media del número de cortes de utilizar SG=$\{2,4,8,16,32\}$ en un conjunto de grafos de prueba. En ambos casos los parámetros de enfriamiento son T_i=100, T_{enfr}=0.995. Como se puede observar, MLrMSATS siempre iguala o reduce el número de cortes obtenido por rMSATS en todos los grafos de prueba. En media (AVG), rMSATS obtiene particiones con aproximadamente un 25 % más cortes que las obtenidas por MLrMSATS, lo cual demuestra claramente las ventajas de usar el paradigma multi-nivel.

Un aspecto interesante a analizar es el efecto de aplicar rMSATS en un número diferente de niveles. La Figura 4.5 compara los resultados obtenidos por rMSATS, MLrMSATS$$, MLrMSATS$, y MLrMSATS sobre varios grafos de prueba. El algoritmo rMSATS corresponde a la aplicación de la heurística híbri-

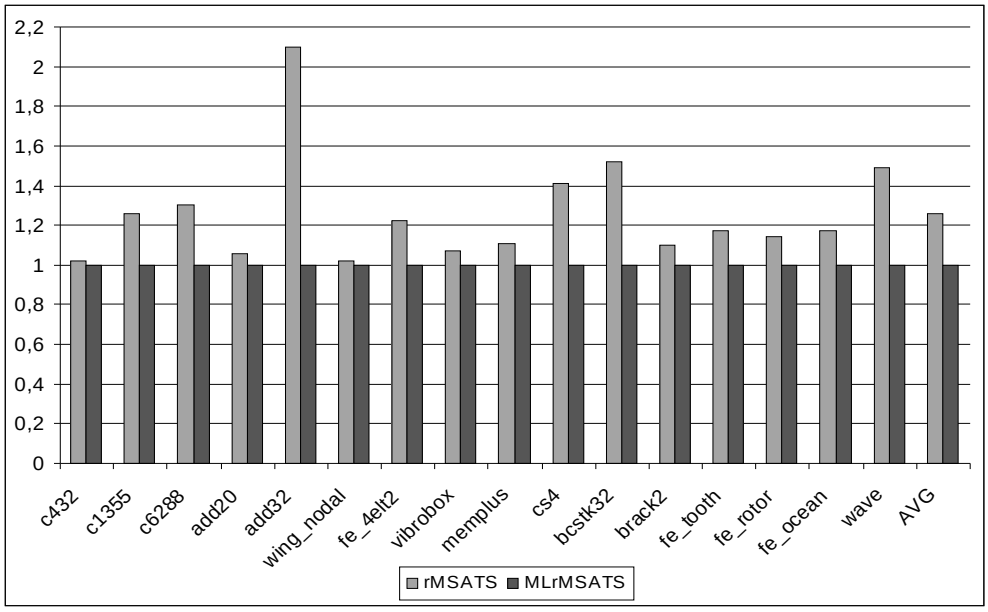

Figura 4.4: Comparativa de MLrMSATS frente a rMSATS.

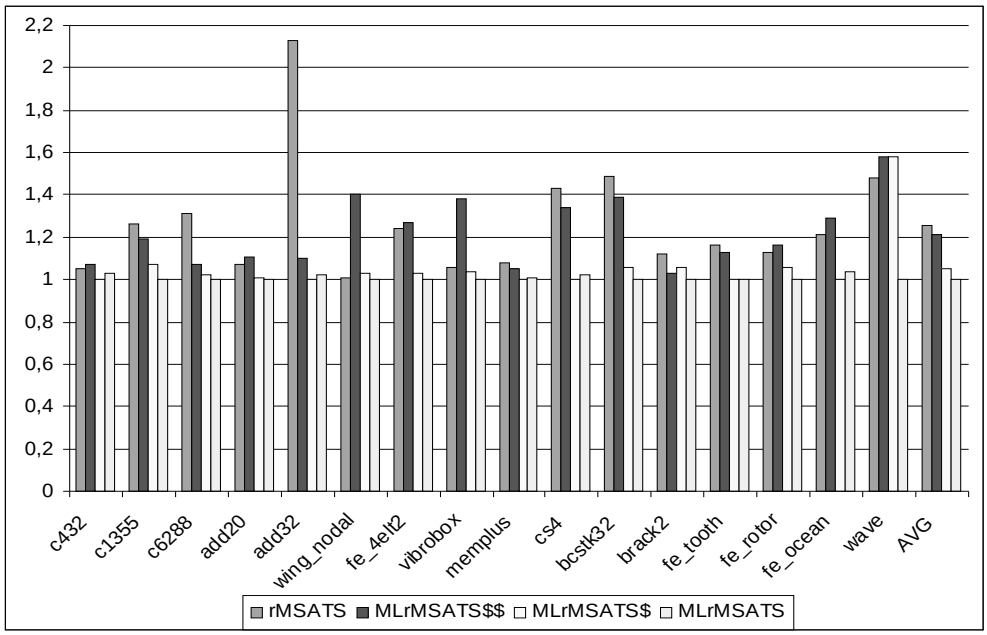

Figura 4.5: Efecto de aplicar rMSATS en diferentes niveles de la fase de refinamiento (número de cortes).

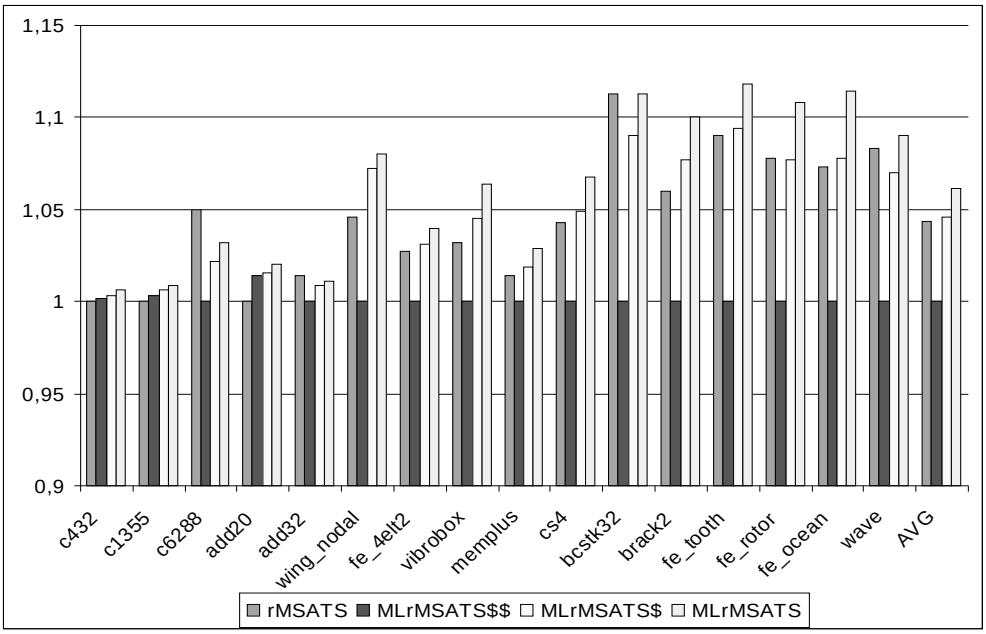

Figura 4.6: Efecto de aplicar rMSATS en diferentes niveles de la fase de refinamiento (tiempo de ejecución).

da solamente en el nivel superior (grafo original). MLrMSATS$$ aplica rMSATS solamente en el nivel inferior y en el intermedio, mientras que en MLrMSATS$ el refinamiento mediante rMSATS se lleva a cabo en el grafo de nivel inferior, alternativamente en los intermedios, y también en el nivel más elevado. Finalmente, MLrMSATS aplica rMSATS en todos los niveles. Los resultados muestran que la calidad de las soluciones obtenidas por MLrMSATS$$ es considerablemente peor que las obtenidas por MLrMSATS$ y MLrMSATS. Por el contrario, MLrMSATS obtiene el menor número de cortes en la mayoría de los casos, y también en término medio (AVG). Por esta razón se puede concluir que la calidad de las particiones obtenidas es directamente proporcional al número de niveles en los que se aplica el procedimiento de refinamiento (rMSATS). Por otro lado, la Figura 4.6 muestra cómo los tiempos de ejecución de esas versiones para los mismos grafos de prueba dependen del número y niveles donde se aplica el refinamiento. Así, por ejemplo, se observa cómo MLrMSATS$$ es el más rápido ya que aplica rMSATS solamente en el nivel inferior e intermedio, mientras que rMSATS, MLrMSATS$ y MLrMSATS lo aplican, entre otros, en el nivel superior (grafo original), lo cual incrementa el tiempo de ejecución. No obstante, el incremento

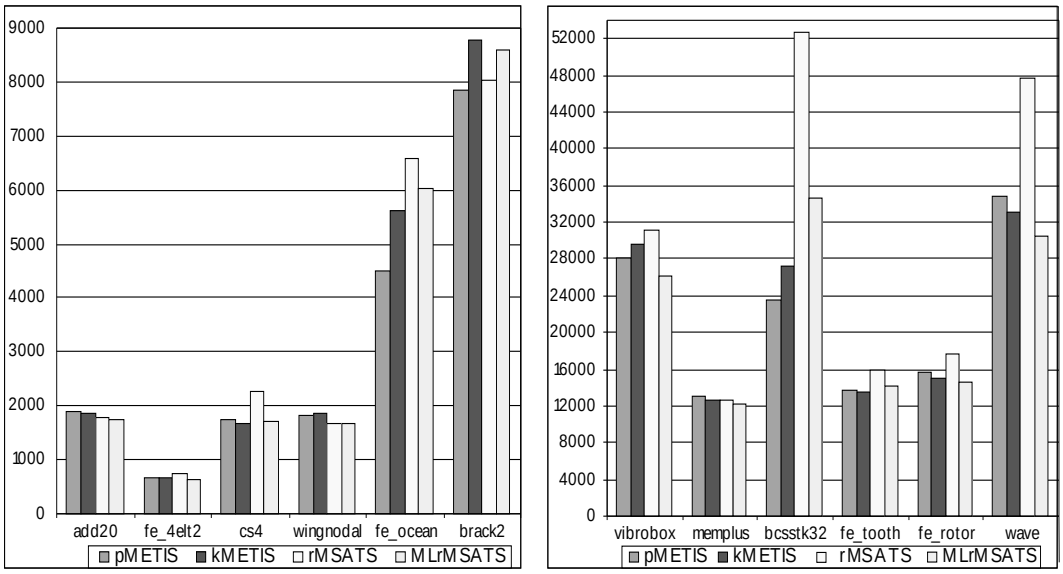

Figura 4.7: Resultados obtenidos en varios grafos de prueba, SG=8.

medio del tiempo de ejecución es inferior al 7 %, lo cual se ve compensado plenamente por la reducción en el número de cortes. No obstante, en función de la aplicación en cuestión, se pueden tomar diferentes decisiones. En aplicaciones dinámicas, donde el tiempo de respuesta debe ser bajo (distribución equilibrada de carga en computadores, etc.), puede ser recomendable aplicar la fase de refinamiento en algunos niveles. Sin embargo, en aplicaciones estáticas, donde se requieren soluciones de alta calidad (diseño VLSI, etc.), resultará conveniente aplicar un refinamiento exhaustivo en todos los niveles.

A continuación comparamos los resultados obtenidos por MLrMSATS frente a METIS [MES], un software de dominio público para repartición de grafos también basado en el paradigma multi-nivel. La Tabla A.6 muestra cómo MLrMSATS mejora a *pMetis* y *kMetis* en la mayoría de los casos, y en el resto se aproxima bastante. Así, por ejemplo, al reparticionar el grafo *vibrobox* en SG=32 sub-grafos, MLrMSATS obtiene una partición con aproximadamente 3500 cortes menos que *pMetis* y 3100 menos que *kMetis*. Esta reducción en número de cortes puede resultar determinante en ciertas aplicaciones reales. Aunque los tiempos de ejecución de MLrMSATS (que oscilan entre varios segundos y pocas horas en función del grafo en cuestión y el número de sub-grafos a dividir) son mayores que en METIS (oscilan entre varios segundos y pocos minutos), esos resultados justifican este aumento en los requisitos computacionales.

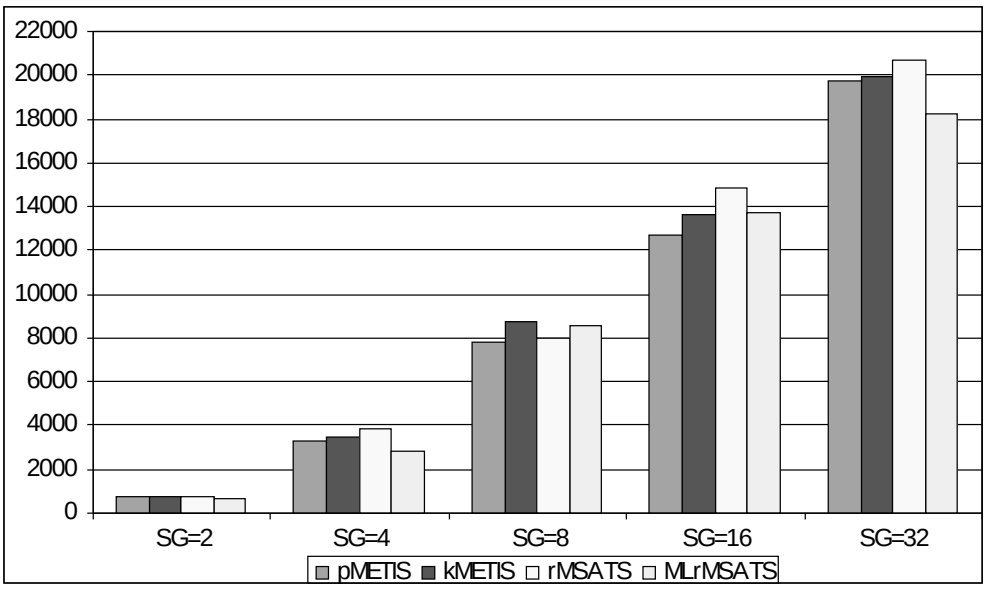

Figura 4.8: Resultados obtenidos en *brack2* en SG={2,4,8,16,32}.

La Figura 4.7 muestra los resultados obtenidos por *pMetis*, *kMetis*, rMSATS, y MLrMSATS al reparticionar diferentes grafos de prueba en SG=8 sub-grafos. Considerando que el eje de ordenadas representa el número de cortes obtenido, se puede observar como MLrMSATS iguala o mejora, en la mayoría de los casos, el resultado obtenido en comparación con los restantes métodos. La Figura 4.8 compara esos algoritmos al reparticionar *brack2* en SG={2,4,8,16,32} sub-grafos, y muestra cómo, de nuevo, MLrMSATS es el que obtiene las mejores particiones en la mayoría de los casos. Así pues, los resultados obtenidos demuestran que el procedimiento multi-nivel MLrMSATS mejora, no sólo a la heurística híbrida rMSATS, sino también a otros métodos encontrados en la literatura.

Tabla 4.1: Tiempos de ejecución (seg.) de rMSATS y MLrMSATS para algunos grafos de prueba.

	c432	*add20*	*wing_nodal*	*bcsstk32*	*wave*
rMSATS	9.6	29.3	138.7	291.0	712.7
MLrMSATS	9.8	30.0	150.2	324.7	776.8

La Tabla 4.1 muestra una comparativa entre los tiempos requeridos por rMSATS y MLrMSATS para particionar algunos grafos de prueba en SG=2 sub-grafos. Debemos indicar que este tiempo se incrementa, no sólo cuando aumenta

el tamaño del grafo a reparticionar, sino también conforme se incrementa el valor de SG, ya que el número de vértices limítrofes entre dos sub-grafos se incrementa con dicho valor. Como se observa, los tiempos de ejecución requeridos por MLrMSATS son sensiblemente superiores a los requeridos por rMSATS. Sin embargo, la diferencia de tiempo no es muy elevada debido a que la fase de agrupamiento y reparticionamiento inicial son muy rápidas en MLrMSATS. Además, en la fase de refinamiento, el mayor esfuerzo computacional se realiza en el nivel superior, mientras que en los niveles bajos/intermedios se realiza rápidamente al trabajar con un número inferior de nodos, y por tanto con vecindades (número de vértices limítrofes entre dos sub-grafos) más reducidas.

4.3. SSSA: Scatter Search with Simulated Annealing

Tal ya como se describió en el Capítulo 1, la aproximación evolutiva llamada búsqueda dispersa (SS) [GLO77, MAR06] fue propuesta como una meta-heurística para programación entera. SS es una técnica flexible, ya que cada parte de su estructura se puede implementar de diferentes formas y grados de sofisticación. La nueva implementación que proponemos consiste en utilizar SA como método de mejora con el objetivo de escapar de los óptimos locales.

4.3.1. Descripción del Procedimiento

El pseudo-código de este nuevo método, llamado Scatter Search with Simulated Annealing (SSSA), se detalla en el Algoritmo 4.3.1. Los parámetros de entrada son el tamaño de la población principal (P_{tam}), y del conjunto de referencia (PR_{tam}), los parámetros de enfriamiento, así como la condición de parada definida por el número de iteraciones/evaluaciones de la función objetivo. A continuación, se obtienen las soluciones de P haciendo uso del método de diversificación y se van mejorando haciendo uso del método de mejora, que en nuestra implementación corresponde a SA. Tras esto, las soluciones se ordenan en términos de la función objetivo ($P[1]$ es la mejor solución, y $P[P_{tam}]$ la peor). Una vez la población ha sido inicializada y mejorada, el siguiente paso consiste en crear el conjunto de referencia (PR). Esta fase de actualización está basada en incluir las mejores $PR_{tam}/2$ soluciones de la población P en el conjunto de referencia (PR). Las restantes $PR_{tam}/2$ soluciones del conjunto de referencia

se obtienen mediante la inclusión de aquellas soluciones de P cuya distancia euclídea respecto a las soluciones ya incluidas sea mayor.

Algoritmo 4.3.1

Begin SSSA (P_{tam}, PR_{tam}, n_e, T_i, T_{enfr}, T_{parada})
  $P \leftarrow \phi$; $PR \leftarrow \phi$; conta_eval←0;
  Desde Conta_P=1 **hasta** Conta_P=P_{tam}
    P[Conta_P] ← Método_Diversificación();
    Método_Mejora_SSSA(P[Conta_P], T_i, T_{enfr}, T_{parada});
  Ordenar_Soluciones_Calidad(P);
  Desde Conta_PR=1 **hasta** Conta_P=PR_{tam}/2
    PR[Conta_PR] ← P[Conta_PR];
  Ordenar_Soluciones_Distancia(P,PR);
  Desde Conta_PR=(PR_{tam}/2)+1 **hasta** Conta_P=tam
    PR[Conta_PR] ← P[Conta_PR-(tam/2)];
  Hacer
    nuevas_soluciones ← FALSE;
    PR' ← Método_Generación_Subconjuntos();
    Hacer
      Seleccionar el próximo sub_conjunto PR'_i de PR';
      candidata ← Método_Combinación_Soluciones(PR'_i);
      candidata2 ← candidata;
      Método_Mejora_SSSA(candidata2, T_i, T_{enfr}, T_{parada});
      Si (f(candidata2) < f(candidata)) **entonces**
        candidata ← candidata2;
        nuevas_soluciones ← TRUE;
      Eliminar PR'_i de PR';
    Mientras ($PR' \neq \phi$);
  Mientras ((nuevas_soluciones=TRUE) & (conta_eval <n_e));
End SSSA

La siguiente fase del algoritmo consiste en combinar y mejorar el conjunto de referencia, para lo cual se determina un conjunto llamado PR', formado por combinaciones de soluciones pertenecientes al conjunto de referencia PR. La implementación aquí presentada consiste en combinar las soluciones de PR en pare-

jas, de forma que PR'=$\{PR[1]\&PR[2]$, $PR[3]\&PR[4]$,...., PR[tam-1]&PR[tam]$\}$. Cada combinación da como resultado una solución de prueba, la cual se va mejorando mediante el método descrito en el procedimiento 4.3.1. Cuando se han mejorado todas las combinaciones, el proceso se repite hasta que no se permiten más mejoras o se supera el número de evaluaciones previamente establecido (el cual se va incrementando tras cada evaluación de la función objetivo). Debemos indicar que si el usuario requiere ejecutar SSSA durante un número determinado de iteraciones/evaluaciones, se puede reinicializar la población principal, generar un nuevo conjunto de referencia y re-iniciar el procedimiento.

El Algoritmo 4.3.2 describe el funcionamiento del método de mejora utilizado por SSSA y que está basado en SA. Tras inicializar la temperatura se modifica una determinada solución haciendo uso de un movimiento de mejora local. Esta nueva solución se acepta o rechaza haciendo uso del criterio de Metropolis [MET53]. Tras esto, la temperatura se decrementa. Este proceso finaliza cuando la temperatura cae por debajo de un determinado umbral (T_{parada}).

Algoritmo 4.3.2

Begin MetodoMejoraSSSA (*candidata*, T_i, T_{enfr}, T_{parada})
  $t \leftarrow T_i$;
  candidata3 $\leftarrow$ *candidata*;
  Hacer
    candidata4 $\leftarrow$ N(*candidata3*);
    Si Metropolis(candidata3,candidata4,t)=TRUE **entonces**
      candidata3 $\leftarrow$ candidata4;
    $t \leftarrow t \cdot T_{enfr}$;
  Mientras ($t > T_{parada}$);
  Retornar *candidata3*;
End MetodoMejoraSSSA

4.3.2. Análisis Experimental

A continuación presentamos los resultados obtenidos por SSSA [BAÑ06d] en la formulación mono-objetivo del problema del diseño de redes de distribución de agua. Además, se han adaptado otras técnicas encontradas en la literatura con el objetivo de obtener conclusiones robustas. En concreto hemos implementado

enfriamiento simulado (SA) [KIR83], MSATS [GIL96], un método que combina búsqueda local iterada [RAM02] con SA [KIR83] (ILSSA) y un algoritmo genético (GA) recientemente propuesto [REC06].

Con el objetivo de optimizar el diseño de redes de distribución de agua, hemos desarrollado un modelo de computador llamado MENOME (MEta-Heuristic pipe Network Optimization ModEl). MENOME, que es una extensión del modelo GENOME [REC06], integra los optimizadores meta-heurísticos descritos anteriormente, un testador de redes hidráulicas, una interfaz de usuario y un módulo de gestión de base de datos. La interfaz gráfica y los algoritmos han sido programados en el lenguaje de programación Visual-Basic. La gestión de la base de datos ha sido implementada utilizando una base de datos relacional y ActiveX Data Objects (ADO). La Figura 4.9 ofrece un diagrama de flujo general correspondiente al funcionamiento de MENOME.

Para realizar las simulaciones, MENOME utiliza el simulador hidráulico de dominio público EPANET (Version 2.00.07) [ROS00] desarrollado por la División de Recursos Hídricos y Suministros de Agua del Laboratorio Nacional de Investigación para la Prevención de Riesgos (NRMRL), perteneciente a la Agencia para la Protección del Medio Ambiente de los Estados Unidos (USEPA) [USEPA].

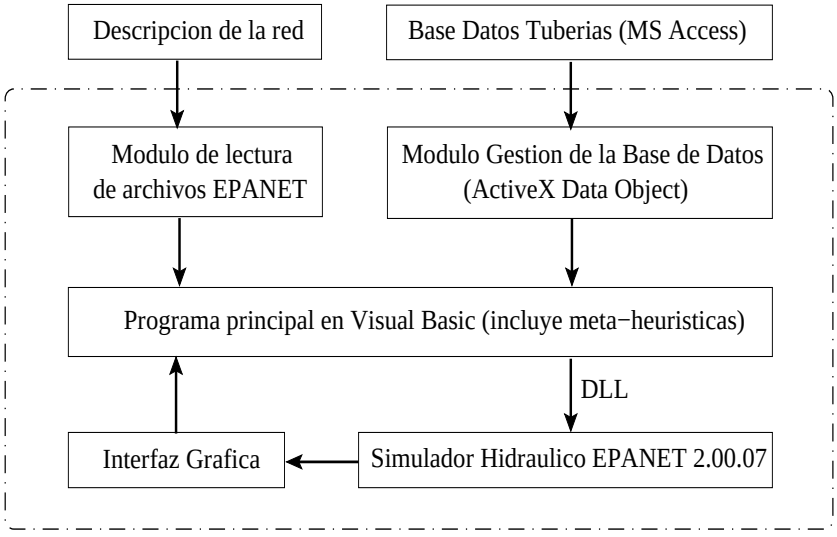

Figura 4.9: Diagrama de flujo de MENOME.

El módulo de herramientas para programadores de EPANET es una librería dinámica de funciones (DLL), que permite a los desarrolladores personalizar el

módulo de cálculo de EPANET para adaptarlo a sus propias necesidades. Las funciones pueden incorporarse en aplicaciones para Windows de 32 bits escritas en C/C++, Delphi, Pascal, Visual Basic, o cualquier otro lenguaje que permita la llamada a funciones incorporadas a una DLL de Windows. Como tal, es un software de dominio público que puede descargarse y distribuirse libremente desde el sitio web. EPANET proporciona un entorno integrado bajo Windows para la edición de los datos de entrada a la red, la realización de simulaciones hidráulicas y de la calidad del agua, así como la visualización de resultados en una amplia variedad de formatos, como mapas de la red codificados por colores, tablas numéricas, gráficas de evolución y mapas de isolíneas. EPANET permite realizar simulaciones en periodos prolongados del comportamiento hidráulico y de la evolución de la calidad del agua en redes de suministro a presión. Una red puede estar constituida por tuberías, nudos (uniones de tuberías), bombas, válvulas y depósitos de almacenamiento o embalses. EPANET lleva a cabo un seguimiento de la evolución de los caudales en las tuberías, las presiones en los nudos, los niveles en los depósitos, la concentración de las especies químicas presentes en el agua a lo largo del periodo de simulación, discretizado en múltiples intervalos de tiempo. Además, puede también simular el tiempo de permanencia del agua en la red y su procedencia desde las diversas fuentes de suministro. Resumiendo, las principales prestaciones del simulador hidráulico de EPANET son:

- no existe límite en cuanto al tamaño de la red que puede procesarse;
- las pérdidas de carga pueden calcularse mediante las fórmulas de Hazen-Williams, Darcy-Weisbach, o Chezy-Manning [ROS00, MOT05];
- contempla pérdidas menores en codos, accesorios, etc.;
- admite bombas de velocidad fija o variable;
- puede calcular el consumo energético y sus costes;
- permite considerar varios tipos de válvulas, tales como válvulas de corte, de retención, y reguladoras de presión o caudal;
- admite depósitos de geometría variable, esto es, cuyo diámetro varía con el nivel;
- permite considerar diferentes tipos de demanda en los nudos, cada uno con su propia curva de modulación en el tiempo;
- puede modelar tomas de agua cuyo caudal dependa de la presión (p.ej. rociadores);
- admite leyes de control simples basadas en el valor del nivel en los depósi-

tos o en la hora prefijada por un temporizador, y leyes de control más complejas basadas en reglas lógicas.

Para que MENOME pueda realizar las simulaciones en conjunción con EPANET se necesitan dos archivos de entrada, uno con la configuración de la red, descrita en el formato de entrada de EPANET, y un segundo archivo correspondiente con la base de datos de tuberías, que es incluida en una base de datos relacional que deben incluir dos campos: el diámetro de cada tubería disponible, así como el coste por unidad de longitud.

Ajuste de Parámetros

Al igual que en el resto de estudios comparativos, resulta imprescindible ajustar los parámetros en función de sus características particulares. La condición de parada de los experimentos no puede establecerse fijando un número de iteraciones fijo, ya que los métodos basados en poblaciones (SSSA, GA) requerirían más tiempo de ejecución que los métodos no poblacionales (SA, MSATS, ILSSA). Dada esta premisa, la mejor forma de garantizar la igualdad de condiciones en la comparación es que todos los métodos realicen el mismo número de evaluaciones de la función objetivo. Este número de evaluaciones, n_e, dependerá de la complejidad del espacio de búsqueda, o lo que es lo mismo del tamaño de la red, la cual viene determinada por el número de enlaces o tuberías (n_l) y el número de posibles diámetros de tuberías (n_d). La Ecuación 4.1 describe como se realiza este cálculo. Considerando un coeficiente multiplicativo K_m=1000, el número de evaluaciones de la función objetivo en la red de Alperovits-Shamir es 9169, 26457 en Hanoi, mientras que en Balerma es 454000.

$$n_e = K_m * n_l * log_{10}(n_d) \tag{4.1}$$

Como se ha comentado anteriormente, la elección de las soluciones iniciales en los métodos heurísticos es un factor muy importante. Con el objetivo de establecer una adecuada comparación entre métodos y evitar en la medida de lo posible la aleatoriedad del proceso de búsqueda provocado por el uso de diferentes soluciones iniciales, todas ellas se han obtenido tomando las tuberías de mayor diámetro para todos los enlaces de la red, lo cual además garantiza que se cumplen las restricciones de presión en los nodos. Por otro lado, y con el objetivo de incrementar la robustez de la comparativa, hemos realizado un

análisis de sensibilidad paramétrica de forma que cada meta-heurística se ha ejecutado 10 veces haciendo uso de diferentes parámetros para la red pequeña (Alperovits-Shamir) y mediana (Hanoi). Tras ello, las mejores configuraciones se han utilizado para la red de mayor tamaño (Balerma). Para las técnicas basadas en enfriamiento simulado (SSSA, MSATS, SA, ILSSA) se han utilizado diferentes esquemas de enfriamiento en cada una de esas diez ejecuciones. MSATS incluye una lista tabú con memoria de 5 configuraciones previas (ψ=5). ILSSA utiliza un factor de distorsión entre cada ejecución de φ=0.25, lo cual implica que entre una ejecución y la siguiente se modifica el 25 % de las tuberías de la configuración anterior. El GA utiliza diferentes probabilidades de cruce y mutación en cada ejecución. La Tabla 4.2 resume los parámetros particulares utilizados en cada meta-heurística.

Tabla 4.2: Parámetros utilizados en Alperovits-Shamir y Hanoi.

GA	P_{tam}	100
	n_i	$f(n_e)$
	ρ_{cruce}	{1.00,0.95,0.90,0.85,0.80,0.75,0.70,0.65,0.60,0.55}
	ρ_{mut}	{0.00,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45}
SA	T_i	{500,250,150,100,75,50,25,10,5,2}
	T_{enfr}	$f(n_e, T_i, T_{parada})$
	T_{parada}	0.01
ILSSA	T_i	{500,250,150,100,75,50,25,10,5,2}
	T_{enfr}	$f(n_e, T_i, T_{parada})$
	T_{parada}	0.01
	φ	0.25
MSATS	T_i	{500,250,150,100,75,50,25,10,5,2}
	T_{enfr}	$f(n_e, T_i, T_{parada})$
	T_{parada}	0.01
	ψ	5
SSSA	P_{tam}	100
	PR_{tam}	10
	T_i	{500,250,150,100,75,50,25,10,5,2}
	T_{enfr}	$f(n_e, T_i, T_{parada})$
	T_{parada}	0.01

Resultados y Discusión

La Tabla 4.3 muestra el coste medio y mínimo obtenido por cada algoritmo en la optimización de la red de Alperovits y Shamir. Como podemos ver, todos los métodos son capaces de obtener el coste de 419000 unidades monetarias. Sin embargo, si calculamos el resultado medio de las diez ejecuciones realizadas podemos observar como SA es el que obtiene el mejor resultado (420100 unidades monetarias), aunque los restantes métodos obtienen resultados con incrementos ($\triangle$) superiores al 1.428 % en todos los casos. En la Figura 4.10 mostramos las líneas de tendencia de cada algoritmo utilizando la mejor configuración paramétrica. Indicar que no mostramos los resultados de ILSSA ya que cada una de sus ejecuciones utiliza el mejor resultado de la ejecución anterior, mientras que esta figura sólo muestra la mejor ejecución de cada método. Como podemos ver todos los métodos convergen a la misma solución (419000 unidades monetarias), aunque los métodos basados en SA convergen más rápidamente.

Tabla 4.3: Resultados obtenidos en Alperovits-Shamir (tras diez ejecuciones).

	coste med.	$\triangle$	coste min.	$\triangle$	mejor config.
GA	423200	1.00738	**419000**	1	0.95/0.05
SA	**420100**	1	**419000**	1	Ti=50
ILSSA	424900	1.01143	**419000**	1	Ti=50
MSATS	424300	1.00999	**419000**	1	Ti=50
SSSA	426100	1.01428	**419000**	1	Ti=50

La Tabla B.1 resume esos resultados para la red de Alperovits-Shamir. Como se observa, las mejores soluciones obtenidas por todas las meta-heurísticas incluidas en MENOME igualan las mejores soluciones encontradas para esta red (419000 unidades monetarias) si no se permite la sub-división y se utilizan los mismos parámetros de Hazen-Williams descritos en el Capítulo 3. Debemos indicar que no todos los resultados son totalmente comparables. Por ejemplo, Eiger y col. [EIG94] encontraron una configuración de 402352 unidades monetarias, pero considerando que cada tubería podría ser dividida a su vez en dos tuberías de diámetros diferentes. Como es de esperar, esta divisividad en las tuberías permite obtener soluciones de menor coste (ver Tabla B.1). Sin embargo, tal y como indicaron Savic y Walters [SAV97], esta sub-división de tuberías no es realista. Otros algoritmos también obtienen menores costes de diseño debido a que utilizan diferentes valores de la ecuación de Hazen-Williams. Más información sobre los métodos que aparecen en la Tabla B.1 pueden encontrarse en [REC06].

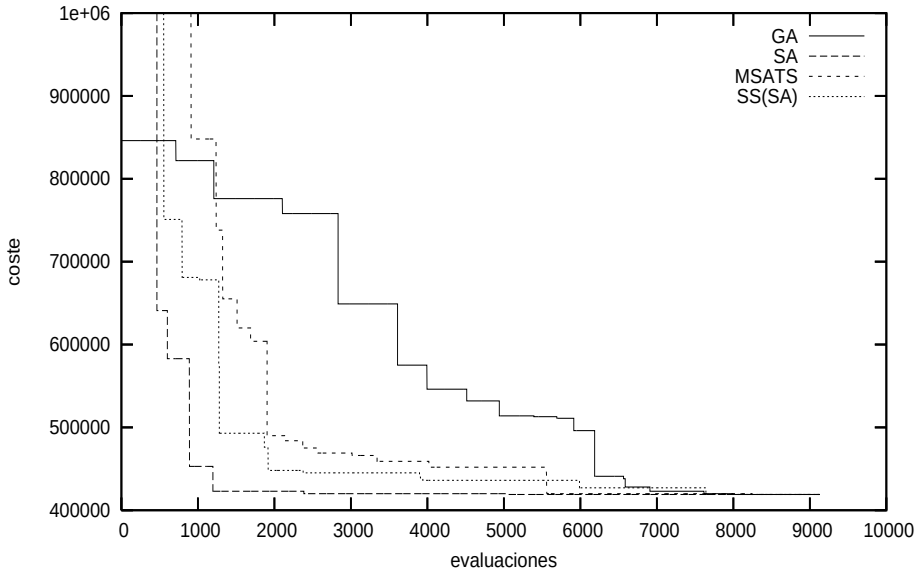

Figura 4.10: Resultados de GA, SA, MSATS, y SSSA en Alperovits-Shamir.

La Tabla 4.4 muestra el coste medio y mínimo obtenido por las meta-heurísticas en la optimización de la red de Hanoi. Los mejores resultados se obtienen mediante el GA (6173421 unidades monetarias), mientras que SSSA también obtiene un buen resultado (6272752). Los restantes métodos obtienen configuraciones sensiblemente más costosas, pero con un incremento por debajo del 3 %. La Figura 4.11 muestra las líneas de tendencia de cada algoritmo utilizando su mejor configuración para Hanoi. En este caso no todos los métodos convergen al mismo resultado, aunque están muy próximos al obtenido por el GA. En particular, SSSA obtiene costes medios del 3.15 % superiores al GA, aunque su valor mínimo mejora al obtenido por SA, ILSSA y MSATS. Al comparar estos resultados con los obtenidos por otros procedimientos previamente propuestos (ver Tabla B.2), se observa como existen soluciones con menor coste, pero de nuevo obtenidas con diferentes parámetros. Así por ejemplo, hay métodos que obtienen costes muy reducidos pero obtenidos al dividir las tuberías o utilizar otros coeficientes de Hazen-Williams. La mejor solución obtenida por un método que use un coeficiente de Hazen-Williams igual a nuestra implementación (C=130) consigue un coste de 6055542 unidades monetarias [SHR01]. Si sólo consideramos soluciones discretas, la mejor solución la obstenta [CUS99], con 6056000 unidades monetarias. Sin embargo, para esos casos los parámetros utilizados

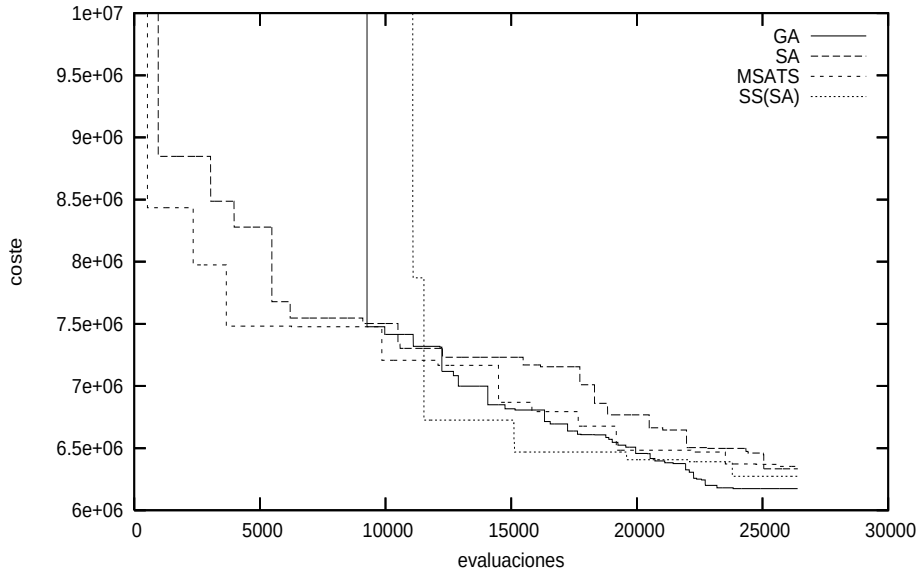

Figura 4.11: Resultados obtenidos por GA, SA, MSATS, y SSSA en Hanoi.

en la ecuación de Hazen-Williams fueron α=10.5088, a=1.85, b=4.87. Savic y Walters [SAV97] llevaron a cabo un análisis de sensibilidad sobre el parámetro de Hazen-Williams, obteniendo soluciones que oscilaban entre 6073000 y 619500 unidades monetarias. La mejor solución utilizando EPANET 2.0 fue conseguida por el algoritmo SFLA [EUS03].

Tabla 4.4: Resultados obtenidos en Hanoi (tras diez ejecuciones).

	coste med.	△	coste min.	△	mejor config.
GA	6575682	1.01415	**6173421**	1	0.95/0.05
SA	**6483950**	1	6333207	1.02588	Ti=25
ILSSA	6510647	1.00412	6308024	1.02180	Ti=50
MSATS	6538453	1.00841	6352526	1.02901	Ti=50
SSSA	6688675	1.03157	6272752	1.01609	Ti=25

Por último, se han evaluado todas las meta-heurísticas en la red de Balerma, utilizando la mejor configuración de las dos redes anteriores. Como podemos ver en la Tabla 4.5, el mejor resultado lo obtiene MSATS (3298268 unidades monetarias), mientras que las restantes meta-heurísticas obtienen un coste superior. El peor en la comparativa es ILSSA, debido a que al ser iterado 10 veces utiliza un esquema de enfriamiento más rápido que SA y MSATS, por lo que

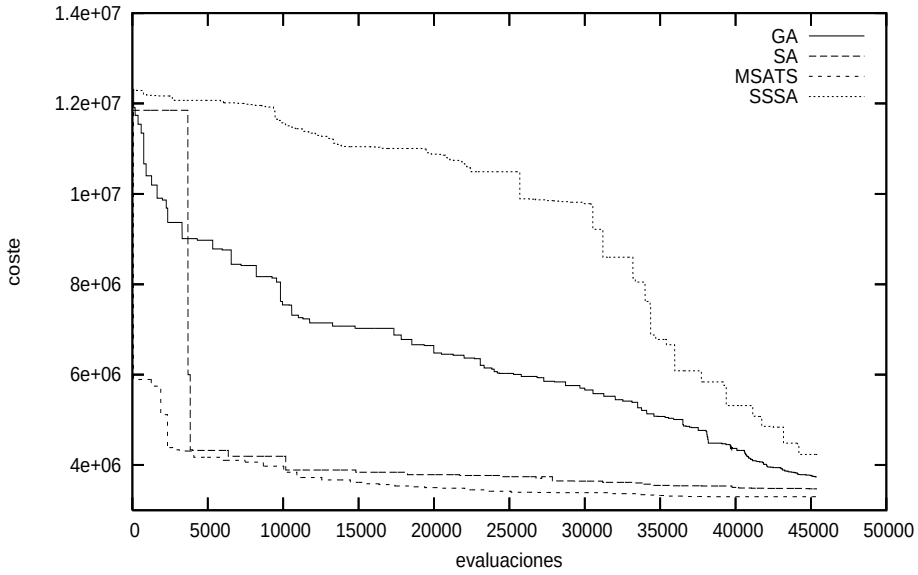

Figura 4.12: Comparativa de GA, SA, MSATS, y SSSA en Balerma.

converge rápidamente a óptimos locales. Como podemos observar en la Figura 4.12, no todos los métodos convergen a la misma solución y se observa como MSATS obtiene el mejor resultado. Debemos indicar que esta red fue recientemente propuesta para evaluar el comportamiento del GA aquí incluido, pero haciendo uso de otros parámetros. Sin embargo, no existen resultados para otros procedimientos, por lo que no es posible establecer más comparativas.

Tabla 4.5: Resultados obtenidos en Balerma (utilizando las mejores configuraciones obtenidas en Alperovits-Shamir, y en Hanoi).

	coste min.	$\triangle$	mejor config.
GA	3737600	1.1332	0.95/0.05
SA	3475740	1.0538	Ti=50
ILSSA	4310016	1.3067	Ti=50
MSATS	**3298268**	1	Ti=50
SSSA	4236030	1.2843	Ti=50

Resumiendo, aunque todos los métodos para todas las redes de prueba obtienen resultados muy aproximados, en problemas de diseño como el que tratamos, esta pequeña reducción porcentual en el coste puede ser decisivo en la toma de decisiones. Si hablamos sobre los tiempos de ejecución podemos indicar que, al

realizar la ejecución en una estación de trabajo de 2GHz con 512Mb de memoria RAM, oscilan entre pocos segundos para la red de Alperovits-Shamir, hasta poco más de una hora para la red de Balerma. Dichos tiempos se pueden observar en la Tabla 4.6, que muestra como los tiempos obtenidos son muy similares en todos los procedimientos, ya que todos ejecutan el mismo número de evaluaciones de la función objetivo. No obstante, existen pequeñas diferencias provocadas por las caraterísticas particulares de las implementaciones. Por ejemplo, se observa como ILSSA incrementa sensiblemente el tiempo de ejecución con respecto a SA, debido a que esta última se ejecuta directamente, mientras que ILSSA necesita modificar la estructura de la red entre cada ejecución independiente, haciendo uso del parámetro φ.

Tabla 4.6: Tiempos de ejecución (seg.) de cada MOMH en las diferentes redes de prueba.

	Alperovits-Shamir	Hanoi	Balerma
GA	26.7	431.1	3721.5
SA	27.2	467.8	3822.2
ILSSA	28.1	485.0	3919.3
MSATS	28.5	492.1	4016.4
SSSA	27.3	479.3	3997.9

4.4. Conclusiones

En este capítulo se han descrito los procedimientos secuenciales mono-objetivo diseñados e implementados en esta tesis doctoral. En concreto se han presentado tres heurísticas, dos de las cuales (rMSATS y MLrMSATS) se han aplicado al problema de repartición de grafos, y otra (SSSA) al problema del diseño óptimo de redes de distribución de agua, ambos descritos en el Capítulo 3. Además, se han adaptado otras técnicas heurísticas encontradas en la literatura con el objetivo de incrementar la robustez de las conclusiones acerca del rendimiento de los procedimientos presentados. En términos generales, se puede decir que la calidad de dichos procedimientos es elevada al ser comparados con otras técnicas previamente propuestas por otros autores. Para el problema de repartición de grafos se ha demostrado empíricamente que el uso de rMSATS dentro del paradigma multi-nivel permite obtener soluciones de mayor calidad, incluso, para algunos grafos de prueba, mejorar a los mejores resultados conocidos hasta la actualidad. En cuanto al problema del diseño óptimo de redes de distribución de agua se ha presentado una heurística híbrida basada en aplicar búsqueda dispersa haciendo uso de enfriamiento simulado como método de mejora. Los resultados obtenidos demuestran que, si bien el procedimiento aquí propuesto no mejora en muchos casos a técnicas propuestas por otros autores, si obtiene resultados próximos a los mismos.

Como acabamos de comentar, en este capítulo se han descrito los procedimientos heurísticos híbridos propuestos para resolver las formulaciones mono-objetivo de ambos problemas. En el siguiente capítulo se presentan otros procedimientos, también híbridos, pero en este caso para resolver las formulaciones multi-objetivo de ambos problemas.

Capítulo 5

Algoritmos Multi-objetivo: Descripción y Resultados

En este capítulo se presentan los procedimientos heurísticos propuestos para resolver las formulaciones multi-objetivo de los problemas descritos en el Capítulo 3. En concreto se presentan dos procedimientos, uno de los cuales (MOSATS) es aplicado a los dos problemas, mientras que MOSSSA se aplica al problema del diseño de redes de distribución de agua. La importancia de este capítulo no radica sólo en presentar dichos procedimientos, sino que además analiza dichos problemas haciendo uso de dos formulaciones multi-objetivo sobre las cuales hay muy pocos trabajos de investigación hasta la actualidad. Además, aunque no se pueden extrapolar directamente conclusiones entre los resultados obtenidos en este capítulo con las obtenidas en las formulaciones mono-objetivo (Capítulo 4), si se puede observar la complejidad que supone trabajar con múltiples objetivos.

5.1. MOSATS: Multi-objective Simulated Annealing and Tabu Search

Como se describió en el Capítulo 2, la hibridación de meta-heurísticas ha sido objeto de un amplio estudio en los últimos años [TAL02]. En el contexto multi-objetivo también se han propuesto métodos híbridos [ISH98, SHI99, KNO00, BUR01, HU03, BAR03], aunque este área de investigación tiene un muchas perspectivas de mejora en el futuro. En esta sección presentamos una nueva meta-heurística para optimización multi-objetivo denominada Multi-Objective Simulated Annealing and Tabu Search (MOSATS) [BAÑ06a]. MOSATS utiliza una población principal de soluciones (P), y otra secundaria correspondiente al archivo externo de soluciones no dominadas (ND) que contiene las soluciones prometedoras encontradas en la población principal. La aceptación de nuevas soluciones en el proceso de búsqueda utiliza SA [KIR83] y TS [GLO93], haciendo uso de relaciones de Pareto-dominancia [GOL89].

5.1.1. Descripción del Procedimiento

Casi todos los MOEAs [DEB00, ZIT01, COR00] utilizan una población de soluciones que se mejoran simultáneamente. Sin embargo, esto es opcional en el caso de las meta-heurísticas basadas en búsqueda local. Como describimos en el Capítulo 2, estrategias basadas en trayectorias como SMOSA [SER93] o UMOSA [ULU99] utilizan una sola solución durante la búsqueda, mientras que otras como PSA [CZY98] o MOTS [HAN97] están basadas en poblaciones. Aunque las versiones no basadas en poblaciones son más rápidas, el uso de diferentes soluciones ayuda a mejorar la dispersión y convergencia de las soluciones no dominadas [RUD00]. Es por ello que el diseño de MOSATS se ha fundamentado en un sistema poblacional. MOSATS comienza aplicando TS en una primera fase, de forma que lleva a cabo movimientos que mejoran la calidad de las soluciones, e introduce dichos movimientos en una lista tabú (LT) intentando evitar ciclos en la búsqueda. Esta lista tabú en MOSATS está formada por una lista de vectores (uno por cada individuo de la población principal), de forma que cada vector almacena la lista de movimientos prohibidos para su solución asignada. Cuando no hay movimientos de mejora, TS pasa el control a SA, la cual hará uso de la información almacenada en LT.

La parte de MOSATS que hace uso de SA trabaja de forma que se aceptarán o rechazarán movimientos utilizando un nuevo criterio basado en masificación,

en combinación con el Criterio de Metrópolis [MET53]. En concreto, esta estrategia divide, para cada solución s, el espacio objetivo en una hiper-malla de 2^K regiones, cuyo punto de referencia es s. Una nueva solución, s', obtenida a partir de s, se situará en una de esas regiones. Si mejora en todos los objetivos se acepta con probabilidad 100 %. Si por el contrario $s \prec s'$, esta se aceptará haciendo del Criterio de Metrópolis, combinando el deterioro en dichos objetivos y la temperatura actual. En el caso de que ambas soluciones sean indiferentes, la probabilidad de aceptación de s' será inversamente proporcional al número de soluciones localizadas en su nueva región (ver Figura 5.1(a)).

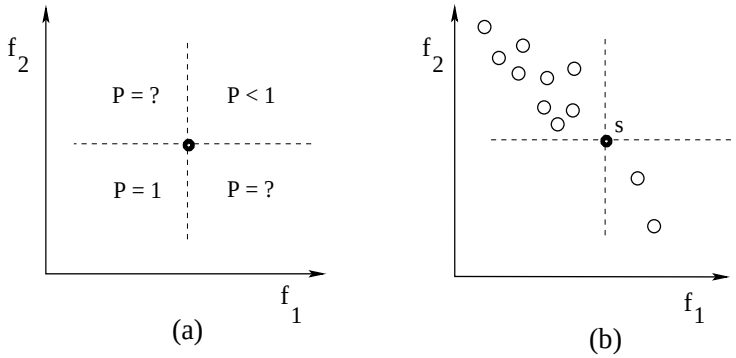

Figura 5.1: Probabilidades de aceptación, K=2 objetivos.

La Tabla 5.1 resume cómo se asignan dichas probabilidades al pasar de la solución s a s' en función del valor actual de t (Pr(s,s',t)). Supongamos que la Figura 5.1(b) muestra la distribución actual de un conjunto de soluciones en un problema con dos objetivos. En este caso, el espacio de soluciones ha sido dividido en 2^2=4 regiones. Como el tamaño de población es P_{tam}=12, es fácil observar que el porcentaje de soluciones en la región superior izquierda es más elevado (9/12) en comparación con la parte inferior derecha (2/12). Así pues, si s' va a esta última región (minimiza f_2 y maximiza f_1), la probabilidad de aceptación es mayor que en el caso de que fuera a la región superior izquierda.

La Figura 5.2 describe el funcionamiento de MOSATS. En la Figura 5.2(a) se muestra la población actual, P. Cada solución lleva a cabo dos fases de búsqueda. La primera, Figura 5.2(b) muestra cómo se mejora una determinada solución haciendo uso de TS. Cuando no hay movimientos de mejora, es decir, nos encontramos en un óptimo local, MOSATS pasa a una segunda fase (Figura 5.2(c)) en la cual se aplica SA con el objetivo de escapar de dicho óptimo local con la aspira-

Tabla 5.1: Reglas de aceptación en MOSATS según las relaciones de dominancia.

Relación	Pr(s,s',t))
$s' \prec s$	1
$s \preceq s'$	$\prod_{k=1}^{K} \min\{1;\ e^{((f_k(s)-f_k(s'))/t)}\}$
$s \sim s'$	$1\text{-}\frac{\text{cardinal}(region(s'))}{P_{tam}}$

ción de alcanzar un área prometedora del espacio de búsqueda. En las primeras etapas de la búsqueda, debido a que la temperatura (t) es elevada, aumenta la probabilidad de aceptación de soluciones que se alejen de óptimos locales, mientras que conforme avance el proceso, y por tanto decrezca la temperatura, irá tomando más importancia la fase de TS que SA. No obstante, debemos indicar que el uso de SA (especialmente cuando el parámetro de temperatura es elevado) presenta como inconveniente el hecho de que se pueden perder soluciones prometedoras durante el proceso de búsqueda. Lo habitual para resolver este inconveniente es utilizar un archivo externo o población secundaria (ND) que almacene las soluciones no dominadas obtenidas por la población inicial [RUD00].

Algoritmo 5.1.1

Begin FAS (ND, s)
Si (ND no esta lleno) **entonces**
  Si ($\nexists s' \in ND : s' \prec s$) **entonces**
    $ND \leftarrow ND \cup s$;
Si no entonces
  Si ($s \prec s'$: $s' \in ND$) **entonces**
    $s' \leftarrow s$;
  Si no entonces
    Si ($\exists k \in K$: $f_k(s) < f_k(s')$, $\forall s' \in ND$) **entonces**
      Si ($\exists s' \in ND : f_{k'}(s') \not\leq f_{k'}(s'')$, $k' \neq k \in K$, $\forall s'' \in ND$) **entonces**
        $s' \leftarrow s$;
End FAS

MOSATS utiliza una nueva estrategia, denominada Fast Archiving Strategy (FAS) [BAÑ06a], que es una simplificación del esquema basado en mallas pro-

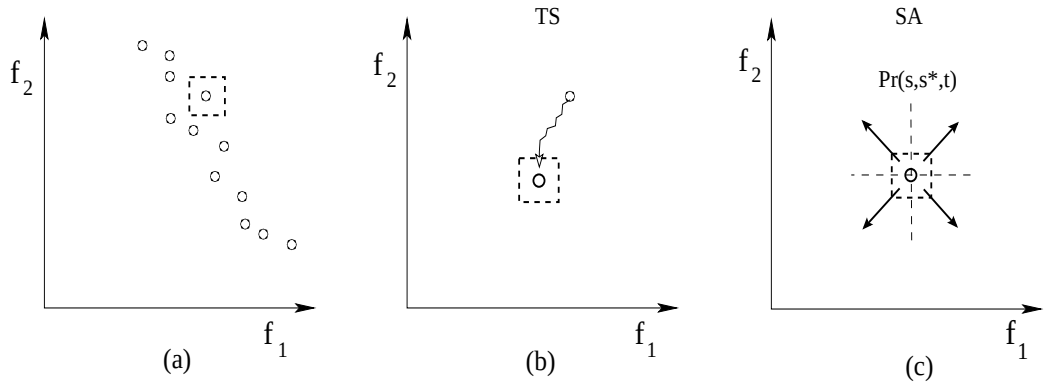

Figura 5.2: Proceso de búsqueda en MOSATS.

puesto en PAES [KNO99]. Como se observa en el Algoritmo 5.1.1, la complejidad algorítmica de FAS es muy reducida. En cada iteración de MOSATS la población P es actualizada considerando las soluciones incluidas en *ND*.

El Algoritmo 5.1.2 ofrece una descripción del funcionamiento de MOSATS. Los parámetros de entrada son los tamaños de la población (P_{tam}) y del archivo externo (ND_{tam}), la temperatura inicial (T_i), el factor de enfriamiento de la temperatura (T_{enfr}), y la condición de parada (T_{parada}). Tras generar las soluciones de la población principal, tanto el archivo externo, como la temperatura inicial son inicializados. El primer paso consiste en actualizar la lista tabú, tras lo cual se inicia un proceso de búsqueda de soluciones no tabú que mejoren la calidad de la solución previa. Si existe dicho movimiento, se lleva a cabo y se incluye en la lista tabú. Cuando no haya movimientos de mejora, el bucle finaliza y se aplica el procedimiento FAS, con lo que la mejor solución encontrada se añade al archivo *ND*. En la siguiente fase se aplica la parte correspondiente a SA. En este caso se obtiene una nueva solución de forma aleatoria. La probabilidad de aceptación de esta nueva solución se calcula tal y como se describe en la Tabla 5.1. Dicho valor se utiliza en la función de Metrópolis [MET53], la cual determinará si se acepta o no. Si se acepta, la nueva solución sustituye a la anterior. El procedimiento FAS se aplica de nuevo, tras lo cual la población principal estará formada por una selección de las mejores soluciones localizadas en la población principal (P) y en el archivo externo (ND). Antes de pasar a la siguiente iteración, el valor de temperatura se decrementa. Al final el bucle externo, se devuelve el conjunto de soluciones no dominadas, *ND*.

Algoritmo 5.1.2

```
Begin MOSATS (P_tam, ND_tam, T_i, T_enfr, T_parada)
  ND←φ;        t←T_i;
  Desde Conta_P=1 hasta Conta_P=P_tam
      P[Conta_P] ← Inicializar();
      LT_s← φ;
  Hacer
  Desde Conta_P=1 hasta Conta_P=P_tam
      Actualizar LT_s;
          Hacer
               detener_búsqueda_tabú←1;
              s'←movimiento(s);
              Si (s'≺s) entonces
                    s←s';
                   detener_búsqueda_tabú←0;
                   LT_s←LT_s∪movimiento(s,s');
          Mientras (detener_búsqueda_tabú=0);
          FAS(ND,s);
          s'←movimiento(s);
          probabilidad←Pr(s,s',t);
          Si (random[0,1]>probabilidad) entonces
               s←s';
              FAS(ND,s);
      S←Selección(S,ND);
      t←t ·T_enfr;
  Mientras (t>T_parada);
End MOSATS
```

5.1.2. Análisis Experimental

A continuación mostramos los resultados obtenidos por MOSATS en la versión multi-objetivo del problema de repartición de grafos. Además, se han adaptado a este problema diversas heurísticas propuestas por otros autores. En par-

ticular, se han implementado dos técnicas basdadas en trayectorias (no poblacionales): SMOSA [SER93], UMOSA [ULU99], y otras dos poblacionales: PSA [CZY98], y MOTS [HAN97], todas ellas descritas en el Capítulo 2.

A continuación describimos las diferentes configuraciones paramétricas utilizada por MOSATS y el resto de las heurísticas con las que es comparada. En cuanto a las soluciones iniciales, todas las MOMHs comparadas utilizan las mismas soluciones iniciales obtenidas mediante el procedimiento FGA [FAH88]. Para todas las MOMHs poblacionales se utilizan tamaños de población principal y externa igual a 100. La lista tabú (LT_s) utilizada en MOSATS consta de 100 vectores de tamaño |V| (número de vértices). Cada vector LT_s corresponde a una solución (individuo) de la población, esto es, una partición candidata, mientras que los elementos en el vector se relacionan con los vértices del grafo. Así pues, si se produce el movimiento de un vértice entre sub-grafos distintos y dicho movimiento se considera *tabú*, su posición en el vector correspondiente a la lista tabú se fija a valor uno. En cada iteración de MOSATS, los vértices tabú (aquellos fijados a uno) no pueden moverse. Cuando se actualiza la lista tabú, los vértices fijados a uno (tabú) en iteraciones previas pasan a tomar el valor cero (no tabú). Los valores de enfriamiento utilizados en MOSATS son: T_i=100, T_{enfr}=0.99, mientras que la condición de parada se satisface cuando la temperatura t cae por debajo del umbral T_{parada}=0.01. El parámetro σ_{cortes} se utiliza en UMOSA para ponderar este objetivo, de forma que σ_{cortes}+$\sigma_{desequilibrio}$=1.

En este estudio comparativo también es necesario establecer un criterio de comparación equitativo entre todos los métodos. Al igual que en otros estudios similares [COR00], es habitual comparar diferentes algoritmos de forma que ejecuten el mismo número de evaluaciones de la función objetivo. En el problema de repartición de grafos cada evaluación es equivalente a visitar un vértice limítrofe entre dos sub-grafos. Por lo tanto, MOSATS se ejecuta haciendo uso de los parámetros de enfriamiento descritos anteriormente. Tras ello, el resto de MOMHs se ejecutan durante el mismo número de evaluaciones. Esto se lleva a cabo en SMOSA y PSA mediante el ajuste de T_{enfr} previamente a su ejecución, de acuerdo al grafo en cuestión. En UMOSA este ajuste consiste en modificar el número de ejecuciones independientes haciendo uso de diferentes pesos en su función objetivo, que en total ejectuarán n_e=1838 evaluaciones. Por último, MOTS también ejecuta dicho número de evaluaciones.

Primeramente, hemos evaluado la mejora de combinar SA y TS frente a ambas técnicas por separado haciendo uso de la métrica de cobertura de conjuntos (SC) descrita en el Capítulo 3. La Tabla 5.2 muestra las relaciones de cobertura

Tabla 5.2: Cobertura de conjuntos (SC) de MOSATS y sus variantes.

	en cada grafo						en media (AVG)		
	MOSATS$_{SA}$		MOSATS$_{TS}$		MOSATS		MOSATS$_{SA}$	MOSATS$_{TS}$	MOSATS
	data	3elt	1.000	0.833	0.000	0.200			
	uk	wing	1.000	0.400	0.429	0.000		0.702	0.108
MOSATS$_{SA}$	vibr.	cti	0.500	0.833	0.000	0.000			
	mem.	cs4	0.714	0.333	0.231	0.000			
	0.000	0.143			0.000	0.000			
	0.000	0.167			0.000	0.000	0.099		0.000
MOSATS$_{TS}$	0.182	0.143			0.000	0.000			
	0.045	0.111			0.000	0.000			
	1.000	0.857	1.000	0.667					
	0.500	1.000	1.000	1.000			0.832	0.786	
MOSATS	0.909	0.571	0.500	0.833					
	0.818	1.000	0.286	1.000					

entre MOSATS y sus versiones simplificadas que solamente utilizan SA o TS en ocho de los grafos de prueba descritos en el Capítulo 3. Además, en la parte derecha de dicha tabla se incluyen los resultados medios de cobertura para dichos grafos. La primera conclusión que obtenemos es que la combinación de SA y TS ofrece una mejora en la calidad de los conjuntos no-dominados. En particular, las soluciones obtenidas por MOSATS dominan al 83.2 % de las soluciones no dominadas obtenidas por la versión que omite TS (MOSATS (SA)), mientras que domina al 78.6 % de las soluciones obtenidas por la versión que solamente utiliza TS (MOSATS (TS)). Por el contrario, las soluciones no dominadas obtenidas por esas versiones no híbridas dominan a MOSATS (SA+TS) en menos del 10.8 % de los casos. Estos resultados demuestran que, al igual que ocurría con el procedimiento mono-objetivo rMSATS, en el contexto multi-objetivo dicha sinergia también resulta efectiva.

A continuación comparamos la versión híbrida frente a las MOMHs descritas anteriormente. La Tabla 5.3 muestra cómo, en media, las soluciones obtenidas por MOSATS mejoran claramente a aquellas obtenidas por las otras MOMHs. En concreto, las soluciones obtenidas por MOSATS dominan al 86.5 %, 65.9 %, 85.9 %, y 95.3 % de las soluciones no dominadas obtenidas por SMOSA, UMOSA, PSA, y MOTS, respectivamente. Sin embargo, esas meta-heurísticas dominan a menos del 54.1 % de las obtenidas por MOSATS. Los resultados para cada grafo

se muestran en la Tabla A.7.

Tabla 5.3: Media de la cobertura de conjuntos (SC) de MOSATS, SMOSA, UMOSA, PSA, y MOTS.

	MOSATS	SMOSA	UMOSA	PSA	MOTS
MOSATS		0.865	0.659	0.859	0.953
MOSA	0.116		0.151	0.315	0.628
UMOSA	0.541	0.709		0.707	0.929
PSA	0.132	0.445	0.083		0.633
MOTS	0.125	0.245	0.173	0.079	

Además de la métrica de cobertura de conjuntos (SC), comparamos dichos resultados haciendo uso de la métrica de hiper-volumen (H) descrita en el Capítulo 3. La Tabla 5.4 muestra los resultados de todos los métodos para los ocho grafos de prueba analizados. Como se observa, MOSATS obtiene el mayor valor de hiper-volumen en muchos casos, y en los que no es así se encuentra muy próximo. Por otro lado, podemos indicar que UMOSA obtiene también buenos valores para esta métrica, siendo el mejor en varios grafos de prueba.

Tabla 5.4: Hiper-volumen (H) de MOSATS, SMOSA, UMOSA, PSA, y MOTS para los grafos de prueba.

grafo	MOSATS	SMOSA	UMOSA	PSA	MOTS
data	**0.5526**	0.3881	0.3010	0.4000	0.0969
3elt	0.7886	0.7197	**0.8026**	0.7889	0.2952
uk	**0.7333**	0.6741	0.5619	0.7313	0.2840
wing_nodal	0.6678	0.4508	**0.6978**	0.4440	0.0082
vibrobox	**0.6198**	0.3768	0.5862	0.4430	0.0372
cti	0.6283	0.4708	**0.7302**	0.6144	0.0764
memplus	**0.4905**	0.4014	0.3477	0.2856	0.0430
cs4	0.7671	0.6015	**0.8027**	0.7103	0.3658

En cuanto a los tiempos de ejecución, podemos indicar que en todos los casos se requieren entre varios minutos y pocas horas de acuerdo al grafo de prueba particionado. Con el propósito de clarificar esos resultados, la Figura 5.3 muestra los conjuntos no dominados obtenidos por todas las MOMHs aquí comparadas, al reparticionar *3elt* en SG=16 sub-grafos. Como podemos ver, las soluciones no dominadas obtenidas por MOSATS dominan a la mayoría de las soluciones

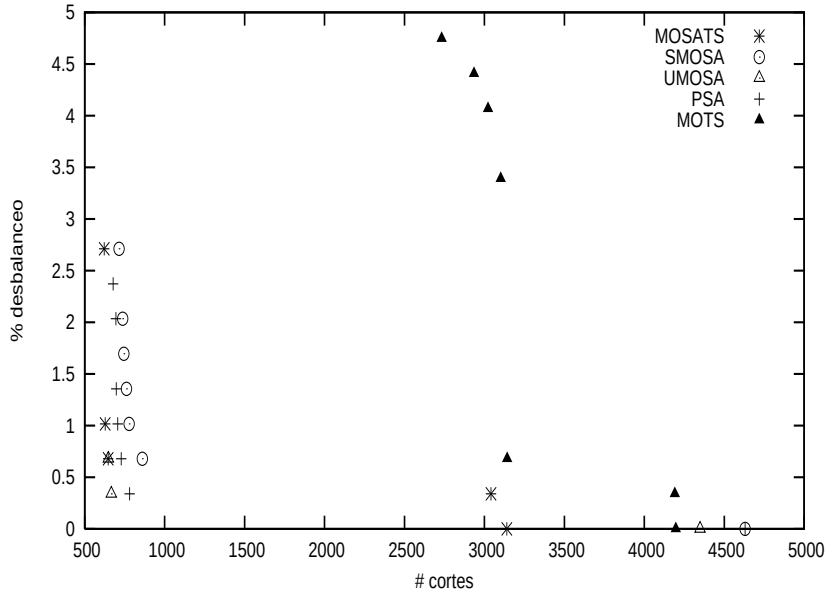

Figura 5.3: Resultados obtenidos en el grafo de prueba *3elt*.

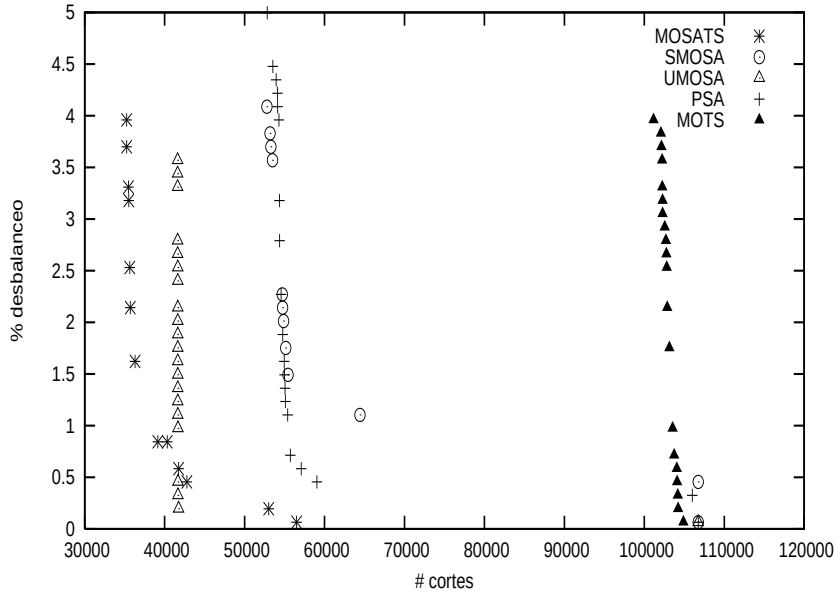

Figura 5.4: Resultados obtenidos en el grafo de prueba *vibrobox*.

obtenidas por los restantes procedimientos, aunque UMOSA es capaz de obtener una solución no dominada por ninguna otra estrategia. Por el contrario MOTS obtiene el peor frente de soluciones. De igual manera, la Figura 5.4 muestra los resultados obtenidos por las MOMHs al dividir el grafo *vibrobox*. Como se puede observar, MOSATS es la que obtiene mejores resultados, aunque UMOSA es capaz de encontrar algunas soluciones que no son dominadas por MOSATS. El resto de métodos obtienen frentes de soluciones de baja calidad.

5.2. MOSSSA: Multi-objective Scatter Search with Simulated Annealing

En el Capítulo 4 se describió SSSA [BAÑ06d], una meta-heurística híbrida que aplicaba SA como método de mejora en la búsqueda dispersa (SS) [MAR06]. Algunos autores han implementado algoritmos basados en SS en el contexto multi-objetivo [NEB05, BEA06, MOL06], aunque es aun un área de estudio abierto a nuevas aportaciones. A continuación proponemos la extensión de SSSA al contexto multi-objetivo.

5.2.1. Descripción del Procedimiento

En esta sección presentamos un nuevo procedimiento, denominado Multi-Objective Scatter Search with Simulated Annealing (MOSSSA) [BAÑ06e], que consiste en extender el funcionamiento estandar de SS en tres aspectos. El primero es que la función objetivo considerará dos objetivos utilizando el concepto de Pareto-dominancia, como en algunas MOMHs descritas anteriormente. El segundo aspecto que incorpora este procedimiento está relacionado con el método de mejora utilizado. En concreto, proponemos aplicar SA con el objetivo de escapar de óptimos locales. La tercera y última extensión incluida en este procedimiento se refiere a que, al igual que métodos como SPEA2, MOSATS, PSA, etc., MOSSSA también incluye un archivo externo de soluciones no dominadas (ND). Este archivo externo es gestionado haciendo uso del procedimiento denominado Fast Archiving Strategy (FAS) descrito en la sección anterior. El pseudo-código de MOSSSA se detalla en el Algoritmo 5.2.1.

Algoritmo 5.2.1

Begin MOSSSA (P_{tam}, PR_{tam}, ND_{tam}, n_e, T_i, T_{enfr}, T_{parada};)
  $ND \leftarrow \phi$; $P \leftarrow \phi$; $PR \leftarrow \phi$; conta_eval←0;
  Desde Conta_P=1 **hasta** Conta_P=P_{tam}
    P[Conta_P] ← Método_Diversificación();
    ND ← FAS(ND,P[Conta_P]);
    Método_Mejora_MOSSSA(P[Conta_P], T_i, T_{enfr}, T_{parada});
  Ordenar_Soluciones_Calidad(P);
  Desde Conta_PR=1 **hasta** PR_{tam}/2
    PR[Conta_PR] ← P[Conta_PR];
  Ordenar_Soluciones_Distancia(P,PR);
  Desde Conta_PR=(PR_{tam}/2)+1 **hasta** PR_{tam}
    PR[Conta_PR] ← P[Conta_PR-PR_{tam}/2];
  Hacer
    nuevas_soluciones ← FALSE;
    PR' ← Método_Generación_Subconjuntos();
    Hacer
      Seleccionar el próximo sub_conjunto $PR'_i \in PR'$;
      candidata ← Método_Combinación_Soluciones(PR'_i);
      candidata2 ← candidata;
      Método_Mejora_MOSSSA(candidata2, T_i, T_{enfr}, T_{parada});
      ND ← FAS(ND,P[Conta_P]);
      Si (candidata2 $\prec$ candidata) **entonces**
        candidata ← candidata2;
        nuevas_soluciones ← TRUE;
      Eliminar PR'_i de PR';
    Mientras $PR' \neq \phi$;
  Mientras ((nuevas_soluciones=TRUE) & (conta_eval <n_e));
End MOSSSA

Los parámetros de entrada corresponden con la planificación de enfriamiento, y los tamaños de la población principal (P_{tam}), del conjunto de referencia (PR_{tam}), y del conjunto de soluciones no dominadas (ND_{tam}), así como el número de evaluaciones (n_e). Tras esto, se inicializan las soluciones de P, y el archivo externo de soluciones no dominadas (ND). Esas soluciones se van mejorando

aplicando el método que describimos en el Algoritmo 4.3.2, tras lo cual se ordenan en términos de la función objetivo ($P[1]$ es la mejor solución, y $P[P_{tam}]$ es la peor). Una vez la población ha sido inicializada y mejorada, el siguiente paso es crear un conjunto de referencia (PR). Esta fase de actualización se basa en incluir las mejores $P_{tam}/2$ soluciones de P en PR. Las restantes $PR_{tam}/2$ soluciones pertenecientes de PR son obtenidas mediante la inclusión de esas soluciones pertenecientes a P cuya distancia euclídea a las soluciones ya incluidas en PR sea mayor.

Algoritmo 5.2.2

Begin MetodoMejoraMOSSSA (*candidata*, T_i, T_{enfr}, T_{parada})
  $t \leftarrow T_i$;
  candidata3 ← movimiento(candidata);
  Hacer
    candidata4 ← N(*candidata3*);
    probabilidad←Pr(*candidata*,candidata3,t);
    Si (random[0,1]>probabilidad) **entonces**
      candidata3 ← candidata4;
    $t \leftarrow t \cdot T_{enfr}$;
  Mientras ($t > T_{parada}$);
  Retornar *candidata3*;
End MetodoMejoraMOSSSA

Tras esto se aplica el procedimiento FAS a todas las soluciones incluidas en PR, incluyendolas en el archivo de soluciones no dominadas (ND). La última fase del algoritmo consiste en combinar y mejorar PR, de forma que se establece un conjunto de combinaciones posibles de soluciones de PR, llamado PR'. En nuestra implementación combinamos esas soluciones en parejas, de forma que PR'=$\{PR[1]\&PR[2],\ PR[3]\&PR[4],....,\ PR[\text{tam-1}]\&PR[\text{tam}]\}$. Cada combinación da como resultado una solución de prueba. Esta solución se optimiza aplicando el método de mejora descrito en el Algoritmo 5.2.2. Tras cada nueva combinación se aplica el procedimiento FAS para almacenar las soluciones en el archivo externo, haciendo uso de la estrategia FAS, propuesta en el procedimiento MOSATS [BAÑ06a]. Este proceso se repite hasta que no se haya obtenido ninguna mejora de ninguna combinación o el número de evaluacio-

nes exceda n_e. En caso de que se haya establecido previamente un número de evaluciones/iteraciones mayor, P puede ser re-inicializado, actualizando PR y continuando la ejecución descrita anteriormente.

5.2.2. Análisis Experimental

A continuación se muestran los resultados obtenidos por MOSSSA en la formulación multi-objetivo del problema de diseño de redes de distribución de agua. Además, se han adaptado algunas de las técnicas descritas en el Capítulo 2 (SPEA2, PESA, PAES, y PSA) y MOSATS.

Al igual que en experimentos previos, aquí todos los procedimientos llevan a cabo el mismo número de evaluaciones de la función objetivo (n_e), según la Ecuación 4.1. Por otro lado, y siguiendo el mismo razonamiento que en el caso mono-objetivo, realizaremos 10 ejecuciones haciendo uso de diferentes parámetros. Así, por ejemplo, SPEA2 y PESA utilizan diferentes probabilidades en sus operadores de cruce y mutación. Para PAES se utilizan diversos número de regiones en la malla que gestiona la población secundaria. Por último, los métodos basados en enfriamiento simulado (PSA, MOSATS y MOSSSA) utilizan diferentes esquemas de enfriamiento. Esos parámetros se describen en la Tabla 5.5, incluyendo la configuración exacta de cada ejecución para clarificar las tablas de cobertura de conjuntos (SC).

Las Tablas B.3,B.4,B.5,B.6,B.7, y B.8 muestran los resultados obtenidos por todos los métodos considerando la métrica de cobertura de conjuntos (SC) en la red de Alperovits-Shamir. Esas tablas están divididas en dos partes. La primera muestra la cobertura entre las diferentes configuraciones paramétricas definidas en la Tabla 5.5, mientras que las últimas dos columnas muestran el resultado medio (AVG) de cada configuración y la desviación típica (D.T.) de todas las configuraciones respecto a la media. Así un valor alto (próximo a 1) en la columna AVG implica que dicha configuración obtiene buenos resultados en términos de cobertura de conjuntos (SC). Por ejemplo, en la Tabla B.3 la mejor configuración es I (0.339) de SPEA2 utilizando ρ_{cruce}=0.60; ρ_{mut}=0.40. Este valor indica que, en media, las soluciones no dominadas obtenidas por la ejecución I son capaces de dominar el 33.9 % de las soluciones no dominadas obtenidas por los restantes métodos. La última columna (D.T.) muestra la desviación típica entre ejecuciones. La Tabla 5.6 resume las tablas previas mostrando solamente los resultados de cobertura de conjuntos media y desviación típica para todas las configuraciones de cada meta-heurística. Esos resultados indican qué configura-

Tabla 5.5: Parámetros utilizados en las ejecuciones multi-objetivo.

	SPEA2/PESA	PAES	PSA/MOSATS/MOSSSA
P_{tam}	100	1	100
ND_{tam}	100	100	100
n_e (ALP)	9190	9190	9190
n_e (HAN)	26400	26400	26400
n_e (BAL)	454000	454000	454000
Ejecución A	ρ_{cruce}=1.00; ρ_{mut}=0.00	2x2	Ti=500
Ejecución B	ρ_{cruce}=0.95; ρ_{mut}=0.05	3x3	Ti=250
Ejecución C	ρ_{cruce}=0.90; ρ_{mut}=0.10	4x4	Ti=150
Ejecución D	ρ_{cruce}=0.85; ρ_{mut}=0.15	5x5	Ti=100
Ejecución E	ρ_{cruce}=0.80; ρ_{mut}=0.20	6x6	Ti=75
Ejecución F	ρ_{cruce}=0.75; ρ_{mut}=0.25	7x7	Ti=50
Ejecución G	ρ_{cruce}=0.70; ρ_{mut}=0.30	8x8	Ti=25
Ejecución H	ρ_{cruce}=0.65; ρ_{mut}=0.35	9x9	Ti=10
Ejecución I	ρ_{cruce}=0.60; ρ_{mut}=0.40	10x10	Ti=5
Ejecución J	ρ_{cruce}=0.55; ρ_{mut}=0.45	11x11	Ti=2

ción paramétrica es la mejor para cada método. Dicha configuración será utilizada para establecer una relación cruzada posterior entre métodos, tal y como se muestra en la Tabla 5.7.

La Figura 5.5 representa gráficamente los frentes de soluciones obtenidos por cada MOMH en la red de Alperovits-Shamir. Se puede observar como la modificación de los parámetros para esta red pequeña tiene un impacto reducido. La única excepción es PAES, donde se observa que el uso de pocas divisiones en la hiper-malla de PAES implica que las soluciones se alejan del frente Pareto-óptimo (desconocido). Sin embargo, una hiper-malla de 9x9 obtiene los mejores resultados. Este valor parece adecuado debido a que valores pequeños de hiper-malla (2x2) no son suficientes para discriminar la dispersión de las soluciones, mientras que valores grandes (11x11) resultan demasiado elevados con respecto al tamaño del archivo externo (ND), lo cual provoca que haya pocas soluciones en cada región. SPEA2 se ve levemente influenciado por los parámetros, mientras que los restantes métodos son independientes de las configuraciones paramétricas. En particular, el uso de diferentes esquemas de enfriamiento en MOSATS y MOSSA no ofrecen una modificación significativa en la calidad de los frentes.

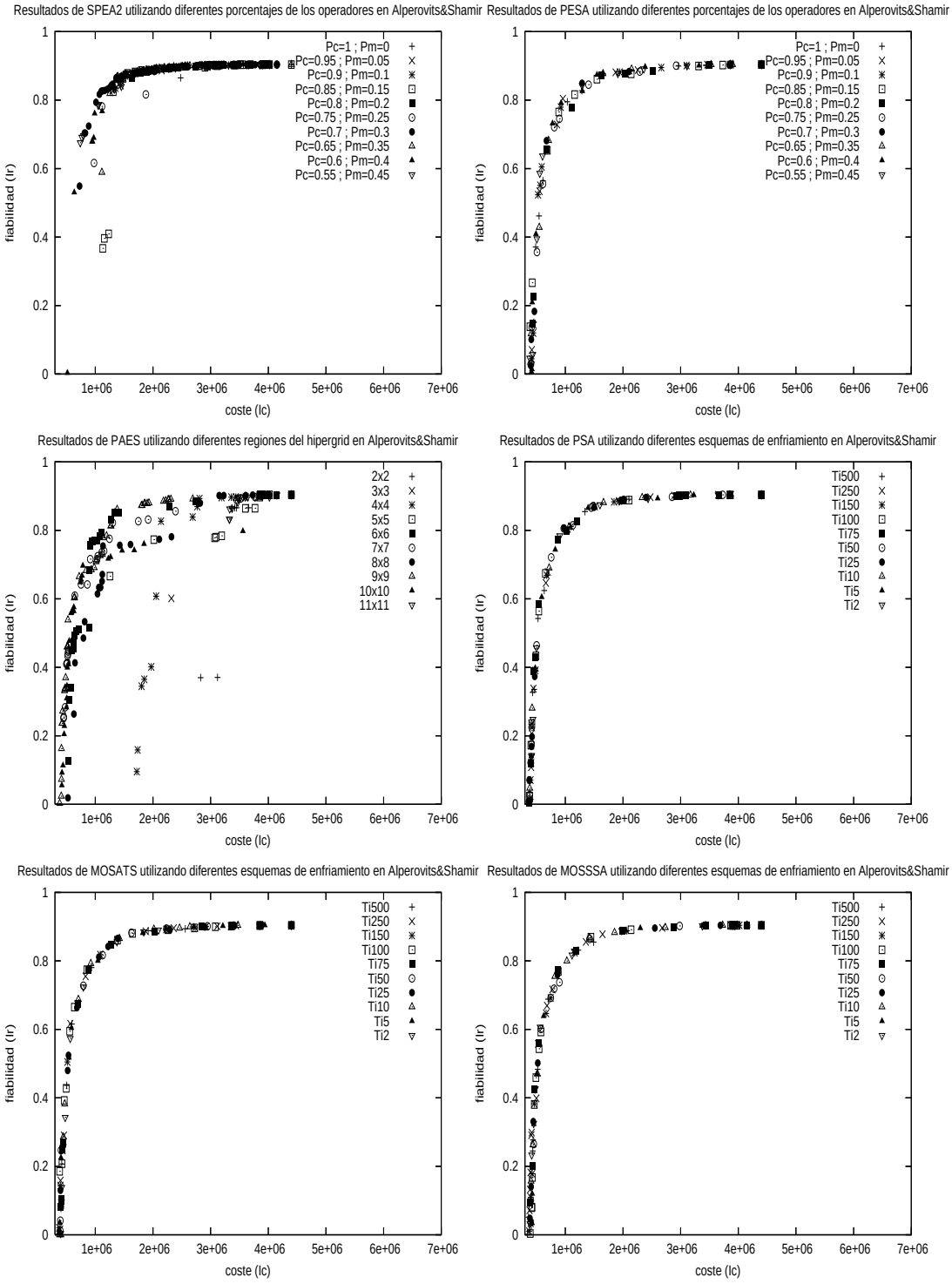

Figura 5.5: Efecto de modificar los parámetros en Alperovits-Shamir.

Tabla 5.6: Resumen de cobertura de conjuntos (SC) y desviación típica en Alperovits-Shamir.

	SPEA2		PESA		PAES		PSA		MOSATS		MOSSSA	
	AVG	D.T.	AVG	D.T.	AVG	D.T.	AVG	D.T.	AVG	D.T.	AVG	D.T.
A	0.000	0.000	0.073	0.059	0.072	0.078	0.065	0.064	0.045	0.087	0.100	0.065
B	0.098	0.145	0.108	0.106	0.028	0.053	0.154	0.078	0.111	0.114	**0.209**	0.062
C	0.256	0.158	0.118	0.079	0.276	0.268	0.080	0.113	0.191	0.097	0.086	0.124
D	0.197	0.128	0.196	0.112	0.162	0.144	0.140	0.106	**0.237**	0.158	0.033	0.037
E	0.225	0.138	0.089	0.093	0.442	0.218	**0.168**	0.119	0.060	0.051	0.123	0.110
F	0.201	0.170	0.184	0.084	0.378	0.193	0.166	0.073	0.159	0.116	0.050	0.059
G	0.312	0.170	**0.211**	0.109	0.432	0.332	0.152	0.094	0.088	0.075	0.134	0.134
H	0.313	0.191	0.137	0.059	**0.494**	0.185	0.061	0.084	0.000	0.000	0.089	0.086
I	**0.339**	0.154	0.164	0.127	0.319	0.178	0.082	0.059	0.213	0.124	0.009	0.024
J	0.303	0.167	0.145	0.103	0.077	0.071	0.092	0.070	0.009	0.024	0.153	0.117

Tabla 5.7: Cobertura de conjuntos (SC) entre MOMHs utilizando las mejores configuraciones en Alperovits-Shamir.

	SPEA2	PESA	PAES	PSA	MOSATS	MOSSSA
SPEA2		0.125	0.435	0.154	0.250	0.133
PESA	0.067		0.109	0.077	0.063	0.133
PAES	0.111	0.125		0.000	0.125	0.000
PSA	0.178	0.500	0.435		0.313	0.067
MOSATS	0.200	0.500	0.587	0.154		0.267
MOSSSA	0.111	0.375	0.370	0.077	0.125	

Los resultados mostrados en la Tabla 5.7 indican que todos los métodos obtienen resultados aproximados, aunque MOSATS, MOSSSA, PSA y SPEA2 son sensiblemente mejores que PESA y PAES. Por ejemplo, MOSATS cubre el 58.7 % de las soluciones obtenidas por PAES, mientras que PAES sólo cubre el 12.5 % de las obtenidas por MOSATS. Esta tabla también muestra como las soluciones no dominadas obtenidas por MOSSSA y PSA son, en su mayoría, indiferentes, ya que las soluciones de PSA dominan solamente el 6.7 % de las soluciones no dominadas proporcionadas por MOSSSA, mientras que MOSSSA cubre el 7.7 % de las soluciones no dominadas obtenidas por PSA. Es evidente la dificultad de establecer un ranking entre métodos, aunque el análisis de cobertura de conjuntos, así como la visualización de las gráficas de los frentes obtenidos

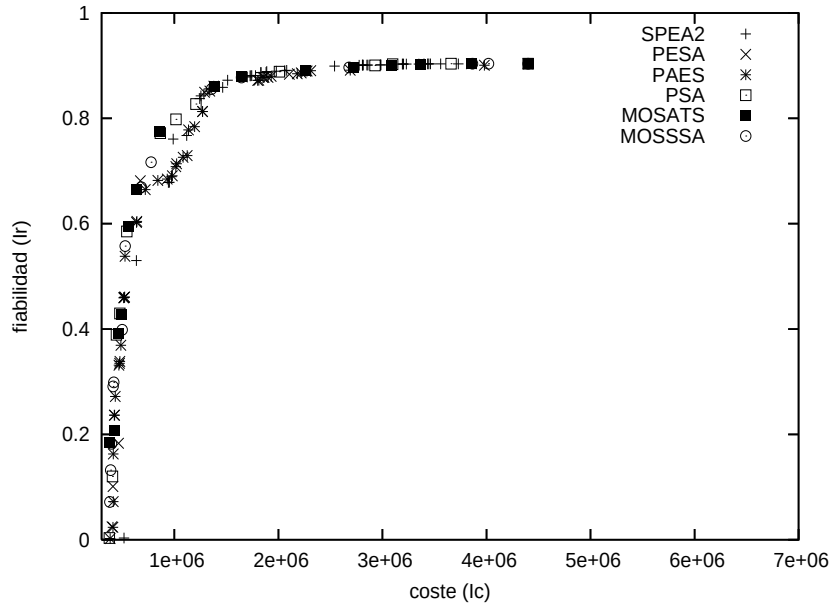

Figura 5.6: Frentes obtenidos por las MOMHs en Alperovits-Shamir.

ayudan a evaluar el rendimiento de dichas MOMHs.

La Figura 5.6 muestra la comparativa utilizando la mejor configuración de cada MOMH en la red de Alperovits-Shamir. Se puede observar como todos los métodos obtienen resultados bastante similares. Por lo tanto, queda claro que todas las MOMHs, incluidas MOSATS y MOSSSA, obtienen resultados muy aproximados en esta red de pequeña escala.

Una vez hemos realizado el análisis detallado de los resultados para la red de Alperovits-Shamir, realizamos lo propio con la red de Hanoi, de tamaño intermedio. Las Tablas B.9,B.10,B.11,B.12,B.13, y B.14 muestran la cobertura media y la desviación típica respecto a esa media por las MOMHs utilizando diferentes configuraciones paramétricas en la red de Hanoi. La Tabla 5.8 resume los datos medios de las tablas previas. Esos resultados muestran cuáles son los mejores parámetros para cada método, que se utilizan para establecer relaciones cruzadas entre MOMHs. Así, la Tabla 5.9 muestra cómo SPEA2 es el mejor método, ya que sólo MOSSSA y MOSATS son capaces de dominar a algunas de sus soluciones, en concreto al 1.9 %, mientras que las soluciones de SPEA2 dominan a más del 56.7 % de las soluciones de las restantes MOMHs. MOSATS y MOSSSA mejoran a PESA, PAES, y PSA.

La Figura 5.7 representa gráficamente los frentes de soluciones no dominadas

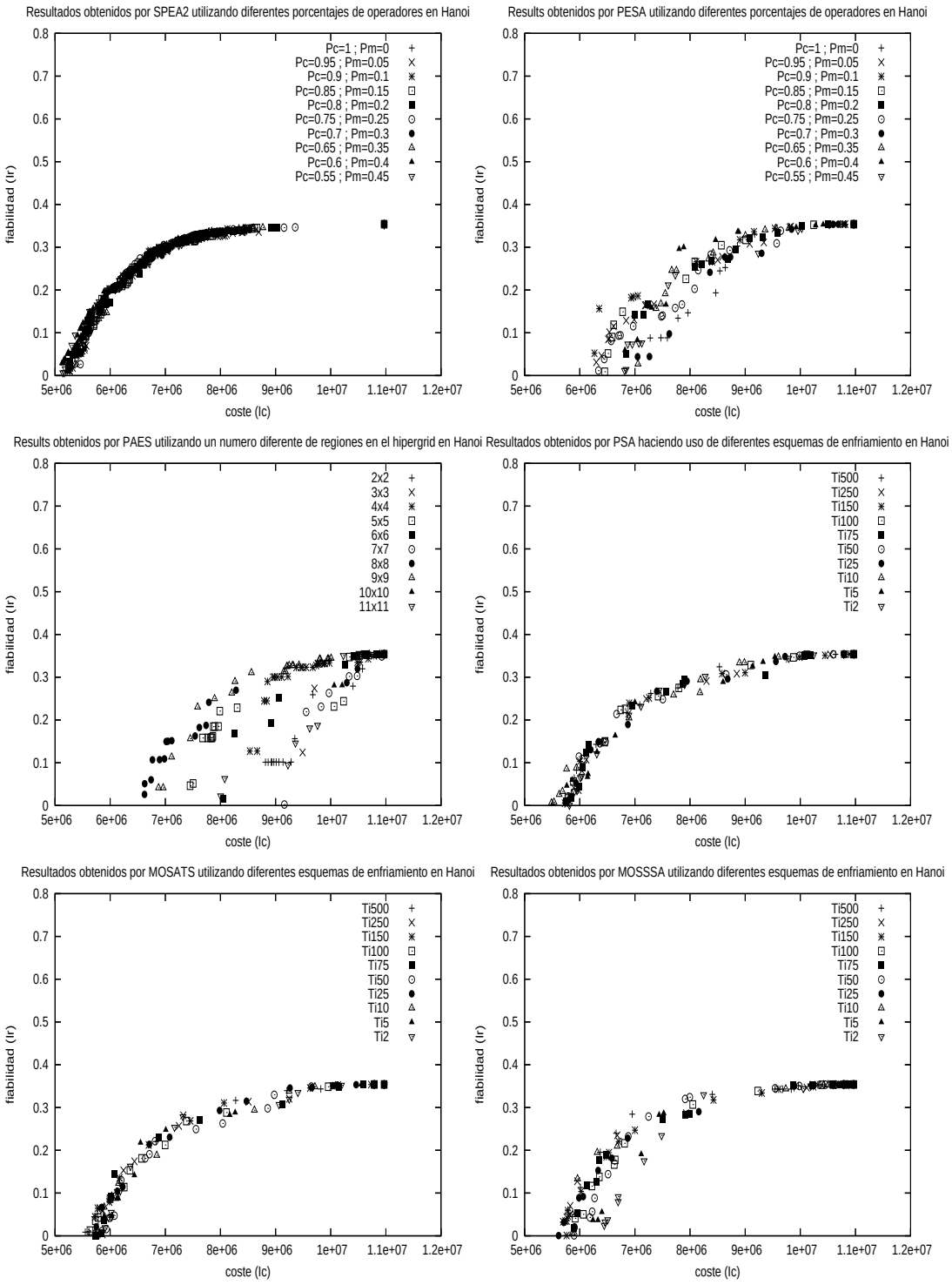

Figura 5.7: Efecto de modificar los parámetros en Hanoi.

Tabla 5.8: Resumen de cobertura de conjuntos (SC) y desviación típica en Hanoi.

	SPEA2		PESA		PAES		PSA		MOSATS		MOSSSA	
	AVG	D.T.	AVG	D.T.	AVG	D.T.	AVG	D.T.	AVG	D.T.	AVG	D.T.
A	0.000	0.000	0.149	0.113	0.159	0.144	0.225	0.138	0.375	0.193	0.436	0.187
B	0.279	0.260	0.466	0.257	0.061	0.117	0.185	0.109	0.251	0.140	**0.468**	0.122
C	0.388	0.272	**0.663**	0.261	0.414	0.343	0.215	0.123	**0.427**	0.202	0.437	0.232
D	0.313	0.280	0.389	0.247	0.379	0.257	0.158	0.075	0.303	0.197	0.225	0.185
E	0.360	0.283	0.245	0.191	0.339	0.285	0.221	0.150	0.243	0.103	0.401	0.226
F	0.457	0.267	0.271	0.235	0.033	0.050	0.250	0.127	0.204	0.099	0.343	0.226
G	0.443	0.263	0.040	0.049	0.601	0.309	0.259	0.116	0.382	0.099	0.370	0.209
H	0.431	0.272	0.285	0.187	**0.699**	0.319	**0.330**	0.156	0.250	0.166	0.405	0.187
I	0.483	0.268	0.525	0.219	0.027	0.042	0.319	0.165	0.213	0.114	0.100	0.158
J	**0.500**	0.257	0.102	0.120	0.299	0.175	0.178	0.090	0.162	0.112	0.096	0.049

Tabla 5.9: Cobertura de conjuntos (SC) entre MOMHs en Hanoi.

	SPEA2	PESA	PAES	PSA	MOSATS	MOSSSA
SPEA2		0.818	0.567	0.733	0.615	0.692
PESA	0.000		0.367	0.067	0.000	0.000
PAES	0.000	0.182		0.000	0.000	0.000
PSA	0.000	0.545	0.767		0.308	0.385
MOSATS	0.019	0.818	0.667	0.333		0.385
MOSSSA	0.019	0.727	0.433	0.200	0.308	

obtenidas por cada MOMH en la red de Hanoi. Como se puede ver, el ajuste de parámetros tiene mayor impacto en este problema que en la red de Alperovits-Shamir, especialmente en PAES, aunque las restantes MOMHs también se ven influenciados por la configuración paramétrica utilizada. Por otro lado, la Figura 5.8 muestra la gráfica comparativa considerando la mejor configuración de cada MOMH en la red de Hanoi. Esta gráfica muestra cómo en esta red, a diferencia de la red de Alperovits-Shamir, la separación entre frentes es significativa. SPEA2 obtiene el mejor frente de soluciones, ya que estas cubren a la mayoría de las soluciones obtenidas por los restantes métodos. MOSATS, MOSSSA y PSA están relativamente cerca, mientras que PESA es sensiblemente peor. Finalmente, todas las soluciones obtenidas por PAES son dominadas por los restantes frentes.

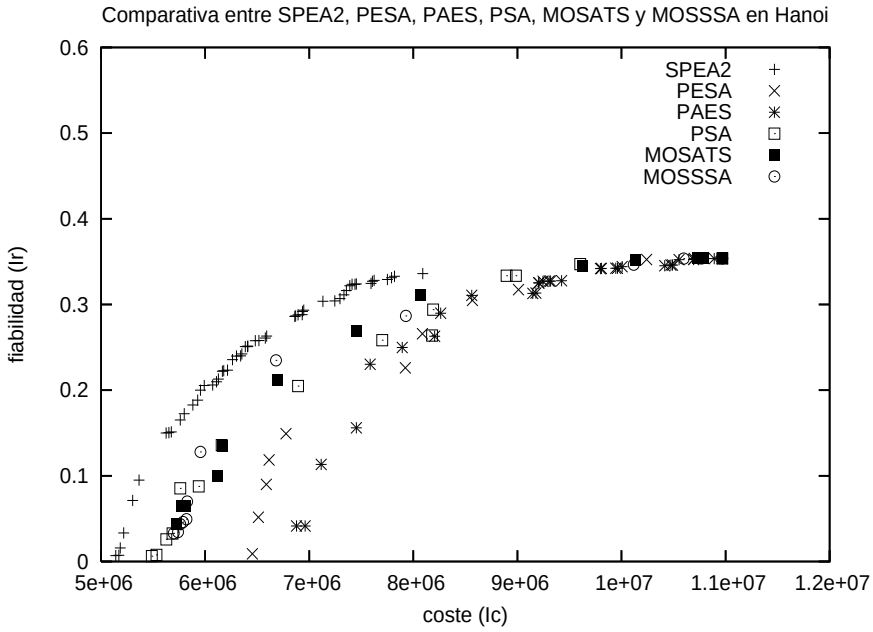

Figura 5.8: Frentes obtenidos por todas las MOMHs en Hanoi.

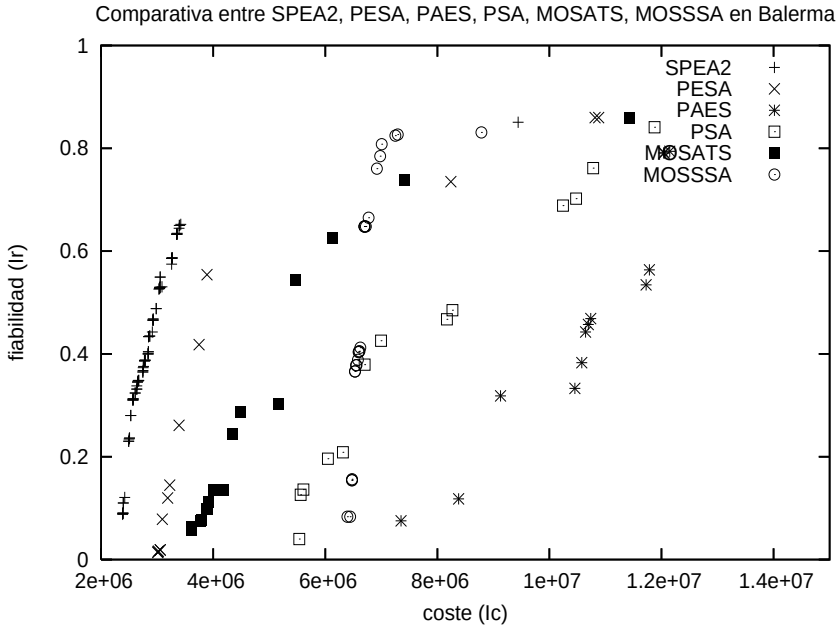

Figura 5.9: Frentes obtenidos por las MOMHs en Balerma.

Aunque no resulta factible establecer conclusiones directas acerca del rendimiento de todas las meta-heurísticas multi-objetivo tomando como referencia aproximaciones mono-objetivo, se pueden comparar en términos de coste. Así, si se comparan esos resultados [REC06] con las MOMHs aquí implementadas se puede ver cómo los costes obtenidos por las aproximaciones multi-objetivo son similares a los previamente obtenidos por las aproximaciones mono-objetivo, tanto para la red de Alperovits-Shamir, como para la de Hanoi, lo cual refuerza las conclusiones de dichas aproximaciones multi-objetivo.

Una vez los parámetros de cada MOMH han sido calibrados en las redes de Alperovits-Shamir y en la red de Hanoi, vamos a evaluar el rendimiento de estos algoritmos en la red de Balerma. En concreto vamos a utilizar las mejores configuraciones compromiso encontradas en la red de Hanoi para optimizar la red de Balerma, ya que Hanoi tiene una escala más parecida a Balerma que la red de Alperovits-Shamir. La Tabla 5.10 presenta los resultados de cobertura de conjuntos (SC) entre MOMHs en la red de Balerma. Debido al gran tamaño y complejidad de esta red, las diferencias entre métodos son aun más significativas que en las redes anteriores. Como se observa, SPEA2 es claramente el mejor método, como lo demuestra el hecho de que las soluciones obtenidas por SPEA2 dominan al menos al 72.0 % de las soluciones no dominadas obtenidas por los restantes métodos, mientras que ninguno de ellos es capaz de dominar a ninguna solución obtenida por SPEA2. PESA, MOSSSA, y MOSATS obtienen soluciones aceptables, sobre todo en lo referente al objetivo de fiabilidad. PAES es, de nuevo, el que obtiene el peor resultado, siendo esta diferencia aun mayor que en redes más pequeñas, debido a que el uso de una sola solución durante el proceso de búsqueda resulta más ineficiente cuando el tamaño del problema aumenta. Dichos resultados numéricos pueden ser interpretados de una forma intuitiva observando la Figura 5.9.

Aunque la red de Balerma ha sido analizada en pocos artículos, podemos comparar a modo de aproximación los resultados aquí obtenidos frente a los resultados obtenidos por aproximaciones mono-objetivo [REC06]. Mientras que SPEA2 obtiene un mejor resultado de 2383537 unidades monetarias, el algoritmo genético propuesto por Reca y Martínez [REC06] obtuvo un coste de 2302423 unidades monetarias. El resultado obtenido por SPEA2 es bueno, si consideramos el hecho de que en el estudio de Reca y Martínez se utilizaron 500 individuos y un número de evaluaciones muy superior.

Por último, la Tabla 5.11 muestra los resultados de hiper-volumen obteni-

Tabla 5.10: Cobertura de conjuntos (SC) entre MOMHs en Balerma.

	SPEA2	PESA	PAES	PSA	MOSATS	MOSSSA
SPEA2		0.786	1.000	1.000	0.905	0.720
PESA	0.000		1.000	0.923	0.905	0.560
PAES	0.000	0.000		0.000	0.000	0.000
PSA	0.000	0.000	1.000		0.000	0.200
MOSATS	0.000	0.071	1.000	0.923		0.560
MOSSSA	0.000	0.071	1.000	0.538	0.048	

dos por todos los procedimientos aquí analizados. Como se puede observar, los resultados obtenidos son bastante aproximados en la red pequeña, siendo los procedimientos basados en enfriamiento simulado los que mejores resultados obtienen. Sin embargo, en las redes de tamaño intermedio este efecto se anula y en la red grande SPEA2 obtiene el mejor resultado. No obstante, PESA, MOSATS y MOSSSA obienen resultados de calidad aceptable. Por el contrario, PSA y PAES obtienen frentes de soluciones con un hiper-volumen muy inferior a las otras técnicas, por lo que parece evidente que su aplicación no es conviente para resolver problemas de gran escala.

Tabla 5.11: Hiper-volumen (H) obtenido por las MOMHs en todas las redes.

	SPEA2	PESA	PAES	PSA	MOSATS	MOSSSA
Hypervolume (ALP)	0.8067	0.8035	0.8312	0.8332	**0.8416**	0.8319
Hypervolume (HAN)	**0.4281**	0.2948	0.2887	0.3475	0.3558	0.3416
Hypervolume (BAL)	**0.6382**	0.5398	0.1359	0.3113	0.4925	0.4360

5.3. Conclusiones

El presente capítulo ofrece una descripción detallada de las técnicas multi-objetivo propuestas en esta tesis. Dichas técnicas están basadas en combinar heurísticas tales como enfriamiento simulado y búsqueda tabú (MOSATS) por un lado, y enfriamiento simulado y búsqueda dispersa (MOSSSA) por otro. El rendimiento de ambas técnicas ha sido evaluado en los problemas de prueba descritos en el Capítulo 3. Los resultados obtenidos cuando se comparan con otras MOMHs encontradas en la literatura muestran que MOSATS obtiene un rendimiento superior en el problema de repartición de grafos, mientras que en el problema de diseño de redes de distribución de agua malladas obtiene resultados que, si bien no son son los mejores absolutos, si están posicionados en un término medio. Por otro lado MOSSSA es analizado en el problema del diseño de redes de distribución de agua, y sus resultados están a un nivel similar a los de MOSATS. En comparación con otros métodos, MOSATS y MOSSSA están situados en una posición intermedia entre técnicas evolutivas puras como SPEA2 y otras técnicas de búsqueda local como PAES.

Además del diseño e implementación de ambas técnicas, también resulta novedoso el hecho de su análisis y aplicación a dos problemas de optimización de gran complejidad. Adicionalmente, dichas técnicas han sido comparadas frente a otras encontradas en la literatura, por lo que implícitamente se ha establecido una comparativa entre un gran número de técnicas, aspecto no analizado por otros autores hasta este momento.

En los Capítulos 4 y 5 hemos podido comprobar el buen comportamiento de las heurísticas híbridas propuestas y de otros procedimientos encontrados en la literatura para resolver problemas de optimización de gran tamaño. Sin embargo, el uso de heurísticas tiene como principal inconveniente el hecho de que la calidad de las soluciones obtenidas se ve influenciada por los parámetros de entrada utilizados. Esta dependencia viene dada, no sólo del problema a resolver, sino también del tamaño y topología de la instancia en cuestión. Por esta razón, en aquellos casos donde el espacio de búsqueda es complejo y/o muy extenso dichas técnicas pueden resultar inservibles para ofrecer soluciones aceptables en tiempos de ejecución admisibles. Bajo dichas circunstancias, el uso del procesamiento paralelo constituye una buena herramienta para reducir dichos tiempos de respuesta e incrementar la eficiencia del proceso de búsqueda. En el Capítulo 6 se presentan varios procedimientos paralelos cuyo principal objetivo es mejorar la calidad del proceso de búsqueda mediante el uso de procesos cooperativos.

Capítulo 6

Paralelización de Meta-heurísticas

En los Capítulos 4 y 5 hemos presentado diversas heurísticas híbridas de optimización mono-objetivo y multi-objetivo que han sido aplicados a ambas formulaciones de los dos problemas descritos en el Capítulo 3. Aunque, como se ha podido comprobar, dichos procedimientos consiguen obtener soluciones de gran calidad, su proximidad al óptimo global disminuye conforme se incrementa el tamaño del problema a resolver. Este hecho, que ocurre no sólo con estos procedimientos sino también en el resto de métodos aproximados, puede ser atenuado haciendo uso del procesamiento paralelo.

Existe una gran cantidad de problemas, entre ellos los incluidos en las categorias de complejidad NP-duros o NP-completos [GAR79] cuyos espacios de búsqueda son tan extensos que los algoritmos secuenciales ejecutados en un solo procesador suelen no ser eficientes, dados los altos requisitos computacionales y el elevado tiempo requerido para obtener las soluciones. En este contexto, el uso del procesamiento paralelo se convierte en una herramienta de gran utilidad a la hora de explorar espacios de búsqueda extensos. Sin embargo, aunque se realice la paralelización correctamente, el problema sigue siendo NP-completo, por lo que no se podría garantizar que la solución obtenida sea la óptima.

El presente capítulo tiene como objetivo analizar las ventajas del uso del procesamiento paralelo a la hora de mejorar el rendimiento de heurísticas de optimización [CEN97, TAL06]. En particular, las diversas implementaciones paralelas que se presentan aquí están enfocadas, no tanto a reducir el tiempo de ejecución de las versiones secuenciales, sino a incrementar la calidad de las soluciones obtenidas haciendo uso de técnicas cooperativas.

6.1. Introducción al Procesamiento Paralelo

Un computador paralelo puede ser definido como un conjunto de dos o más procesadores que, entre otras cualidades, ofrecen la posibilidad de trabajar cooperativamente para resolver diferentes tareas y problemas [BLA00]. Esta definición incluye desde redes de estaciones de trabajo hasta supercomputadores con cientos de procesadores, pasando por clusters de computadores. En la actualidad, la tendencia apunta hacia los clusters de computadores, ya que dichas plataformas ofrecen una buena relación prestaciones/coste, además de interesantes características de reconfigurabilidad, tolerancia a fallos, etc.

A la vez que se han diseñado nuevas arquitecturas paralelas, se han desarrollado interfaces de programación que facilitan las tareas de implementación en las mismas, como por ejemplo las librerias para programación basadas en paso de mensajes [GEI94, SNI96]. La paralelización con estas herramientas es generada de forma casi íntegra por las decisiones del programador.

6.1.1. Arquitecturas Paralelas más Utilizadas

En esta sección se ofrecen los conceptos básicos del procesamiento paralelo en computadores de altas prestaciones. En concreto se describen brevemente algunas de las arquitecturas paralelas más utilizadas, así como las métricas básicas utilizadas para evaluar el rendimiento de programas paralelos.

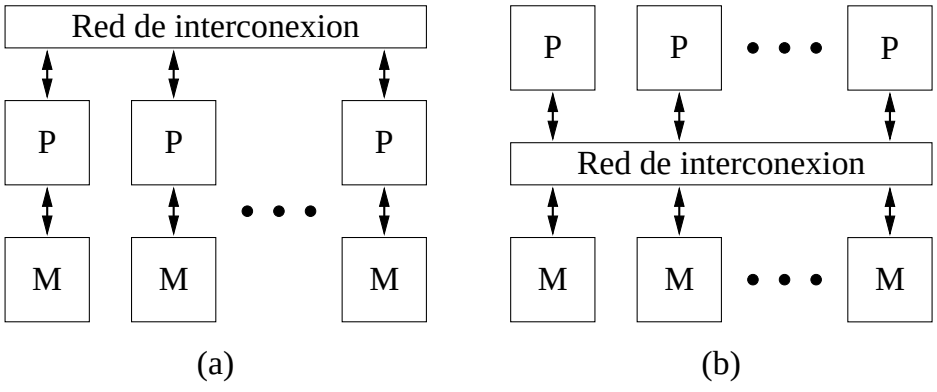

Figura 6.1: Descripción de alto nivel de un (a) multicomputador, (b) multiprocesador de memoria compartida (M=memoria; P=procesador).

Multiprocesadores de Memoria Compartida

Los *multiprocesadores de memoria compartida* proporcionan un espacio de memoria único para todos los procesadores, simplificando la tarea de intercambiar datos entre los procesadores. El acceso a memoria compartida ha sido implementado tradicionalmente usando una red de interconexión entre los procesadores y la memoria. A esta arquitectura se le conoce tradicionalmente como arquitectura UMA (*Uniform Memory Access*), tal y como se observa en la Figura 6.1(b).

La arquitectura paralela prevalente es el multiprocesador de pequeña o moderada escala que proporciona un espacio de direccionamiento global y acceso uniforme a toda la memoria principal desde cualquier procesador. A esta arquitectura se le denomina Multiprocesador Simétrico o SMP [HES02]. Cada procesador tiene su propia caché y todos los procesadores y módulos de memoria están conectados a la misma red de interconexión, usualmente mediante un bus compartido. Los SMPs dominan el mercado de los servidores y son cada vez más comunes. La compartición eficiente de los recursos, como la memoria y los procesadores, así como la posibilidad de acceder a todos los datos compartidos de forma eficiente desde cualquier procesador usando operaciones de lectura y almacenamiento hacen a este tipo de máquinas atractivas tanto para entornos multi-tarea como para la programación paralela.

El elemento de diseño clave de esta arquitectura es la organización de una jerarquía de memoria extendida. Sin embargo, este tipo de arquitecturas paralelas requieren una adecuada gestión de la memoria, con el objetivo de evitar problemas de *coherencia de caché*. Este problema consiste en que cuando un bloque de memoria se encuentra en uno o más procesadores y otro distinto lo modifica, hay que evitar que el resto de procesadores accedan al contenido no actualizado existente en su memoria local. Esto conlleva que sea necesario aplicar técnicas de sincronización para controlar el acceso a las variables compartidas [CUL98].

Bajo este tipo de arquitecturas se utiliza el modelo de programación de memoria compartida, haciendo uso de librerías como OpenMP [OPE98].

Multicomputadores

Una forma de evitar los problemas de los multiprocesadores de memoria compartida consiste en hacer uso de multicomputadores. Los multicomputadores [HAW98] constan esencialmente de un conjunto de procesadores, cada uno conectado a su memoria local. Los procesadores se comunican mediante el paso

de mensajes a través de una red de interconexión. La Figura 6.1(a) muestra un esquema simple de esta arquitectura.

La programación de multicomputadores se realiza mediante librería de paso de mensajes. La complejidad de este modelo de programación radica en que el programador debe planificar la distribución de código y datos entre los diferentes procesadores de manera eficiente, estableciendo la secuencia correcta de envío de mensajes que necesiten otros procesadores [CUL98]. El código para cada procesador se carga en la memoria local al igual que cualquier dato que sea necesario. Los programas están divididos en diferentes partes, como en el sistema de memoria compartida, y dichas partes se ejecutan concurrentemente por procesadores individuales. Cuando los procesadores necesitan acceder a la información de otro procesador, o enviar información a otro procesador, en lugar de acceder a su memoria local, lo adquiere mediante el envio de mensajes. La principal ventaja de esta arquitectura es que es directamente escalable y presenta un bajo coste para sistemas grandes.

Multiprocesadores de Memoria Compartida-Distribuida

Recientemente, los multiprocesadores de memoria compartida han seguido algunas de las tendencias establecidas previamente por los multicomputadores. En particular, la memoria ha sido distribuida físicamente entre los procesadores, reduciendo el tiempo de acceso a la memoria e incrementando la escalabilidad. A estos computadores paralelos se les denominan multiprocesadores de memoria compartida-distribuida (DSM). Los accesos a la memoria remota se realizan a través de una red de interconexión, de una manera similar al caso de los multicomputadores. La diferencia principal entre los DSMs y los multicomputadores es que los mensajes se inician en respuesta a accesos a la memoria en vez de llamadas al sistema. Para reducir la latencia de la memoria, cada procesador tiene varios niveles de memoria caché. Esta arquitectura proporciona un tiempo no uniforme de acceso a la memoria (NUMA) el cual se debe principalmente a la diferencia entre los tiempos de acceso a las cachés y la memoria principal, más que al diferente tiempo de acceso entre las memorias locales y remotas. El Cray-T3E es un ejemplo de esta arquitectura. El principal problema que aparece en los DSMs es la coherencia de caché.

Otra arquitectura de este tipo es COMA (Cache-Only Memory Architecture). En el modelo COMA, cada procesador tiene una parte de la memoria compartida, como en el modelo NUMA, pero cada parte es una memoria caché,

es decir, toda la memoria compartida se comporta como una memoria caché. Por lo tanto, parte de la memoria caché puede tener datos que no han sido demandados por el procesador asociado. La arquitectura COMA permite la migración de datos hacia el procesador que realiza la petición, al igual que en el caso de cualquier multiprocesador con memoria caché. Dado que puede ser costoso que toda la memoria sea una caché, Hagersten y col. en 1991 [HAG91] describen un sistema COMA como una porción de memoria que tiene las funciones de caché, mientras que el resto de la memoria corresponde a lo que ellos denominaron una memoria de atracción (los datos son atraidos a la memoria de atracción más cercana al procesador que los demandó).

6.1.2. Medidas de Rendimiento en Algoritmos Paralelos

En esta sub-sección describimos brevemente algunas de las métricas de rendimiento más utilizadas a la hora de determinar el rendimiento de un algoritmo paralelo [HAW98].

Ganancia de Velocidad o Speedup

La ganancia de velocidad (speedup) se define como:

$$Speedup = S(np) = \frac{T(1)}{T(np)} \tag{6.1}$$

donde np denota el número de procesadores elementales (homogéneos) de la máquina paralela, $T(1)$ el tiempo de ejecución del algoritmo en un solo procesador, y $T(np)$ el tiempo de ejecución para un algoritmo que utiliza np procesadores.

El objetivo del procesamiento paralelo es reducir el tiempo para generar la solución de un problema determinado. Se denomina *ganancia de velocidad lineal* al caso ideal es razonable en el que np procesadores obtengan la solución secuencial en un tiempo np veces inferior al requerido por un solo procesador, es decir:

$$S(np) = \frac{T(1)}{T(np)} = np > 1 \tag{6.2}$$

Sin embargo, en la práctica resulta difícil obtener dicha ganancia de velocidad lineal, y en la mayoría de los casos el valor de $T(np)$ suele ser superior al valor de $\frac{T(1)}{np}$, debido a las penalizaciones relacionadas con la sincronización y las

comunicaciones. No obstante, existen múltiples referencias en las que se muestran valores de la aceleración para algunos programas, por encima del número de procesadores, lo que se denomina ganancia de velocidad super-lineal, es decir,

$$S(np) = \frac{T(1)}{T(np)} > np > 1 \tag{6.3}$$

La razón de esta super-linealidad puede venir motivada por diversos factores, entre los que destacan:

- *Incremento de los recursos en los sistemas paralelos.* Los programas paralelos que exploten el aumento de los recursos de las máquinas paralelas pueden obtener valores de aceleración por encima de los lineales gracias, no sólo a la mayor capacidad de procesamiento en términos de número de procesadores, sino además por el incremento de otros recursos tales como la capacidad de la memoria principal y de la memoria caché.

- *Características propias de los algoritmos paralelos.* Algunos algoritmos paralelos evolucionan de forma distinta dependiendo del número de procesadores. Principalmente esto ocurre en situaciones donde interviene en alguna medida la aleatoriedad. Por ejemplo, en el problema de repartición de grafos tratado en esta tesis se ha comentado que la forma de realizar la repartición inicial haciendo uso de un algoritmo de crecimiento que parte de distintos vértices iniciales permite incrementar la diversidad en la búsqueda, con lo cual se puede alcanzar determinadas soluciones en un tiempo más reducido tomando ventaja de ello.

- *Ineficiencia de los algoritmos secuenciales.* En numerosos casos se realizan implementaciones secuenciales que no son del todo óptimas. Esto conlleva a que dichas implementaciones requieran de elevados tiempos de ejecución para obtener el resultado. Al hacer uso del procesamiento paralelo se puede solventar este handicap de forma que se puede llegar a conseguir una ganancia de velocidad super-lineal.

No obstante, también es conveniente indicar que en muchos casos el uso del paralelismo no ofrece una compensación satisfactoria. Este es el caso en el que el programa paralelizado sea inherentemente secuencial, lo cual provoca que las porciones de código paralelizable sean tan reducidas que su paralelización resulta ineficiente. Este problema viene determinado por la conocida como *Ley de Amdahl* [AMD67].

Eficiencia del Sistema

Otra medida de rendimiento habitualmente utilizada es la *eficiencia del sistema* [LEE80], que representa la ganancia de velocidad por procesador. Así, para un sistema con np procesadores viene definida por:

$$E(np) = \frac{S(np)}{np} \tag{6.4}$$

Como en la mayoría de casos $1 \leq S(np) \leq np$, entonces $1/np \leq E(np) \leq 1$.

6.2. Técnicas de Paralelización de Heurísticas

La programación paralela y distribuida [BLA00] se considera como un mecanismo para acelerar el proceso de búsqueda en la resolución de problemas de optimización de gran tamaño. Además, el uso simultáneo de paralelismo y cooperación permite la mejora en la calidad de las soluciones, que en el caso multi-objetivo implica obtener frentes de soluciones no dominadas más próximas al frente Pareto-óptimo.

La paralelización de meta-heurísticas poblacionales resulta muy eficiente debido a que cada procesador puede trabajar con uno o varios individuos de esas poblaciones. En concreto, el operador de mutación y la evaluación de la función objetivo de cada individuo se puede llevar a cabo en cada procesador de forma independiente. Sin embargo, para otros aspectos como la selección o el cruce de individuos puede ser necesario el intercambio de información entre diferentes procesadores. La paralelización de EAs/MOEAs se ha analizado con detalle en la última década [CAN97, ALB02, VEL03]. A continuación describimos los tres principales esquemas de paralelización más utilizados al paralelizar EAs/MOEAs.

- El modelo *maestro-esclavo* (o single-population master-slave) [CAN97] permite mantener la secuencialidad del algoritmo original. Un procesador maestro centraliza la población y gestiona la selección y los reemplazos de individuos. Este también se encarga del envío de sub-poblaciones a los esclavos, los cuales ejecutan tareas de evaluación, y aplicación de ciertos operadores. Tras esto, los esclavos devuelven las soluciones evaluadas al maestro. Esta aproximación es especialmente utilizada cuando el coste computacional de generar y evaluar nuevas soluciones es elevado.

- El modelo de *islas* (o multiple-population coarse-grained) [CAN97] divide la población original en un conjunto de sub-poblaciones distribuidas entre diferentes procesadores. Cada procesador es responsable de la gestión de la sub-población asignada, de forma que ejecuta todos los pasos de la meta-heurística, y ocasionalmente migra individuos entre islas. Aunque este modelo rompe la secuencialidad del algoritmo, estudios en diferentes aplicaciones han demostrado que suele mejorar al modelo maestro-esclavo, debido principalmente a que permiten mantener la diversidad en la búsqueda a la vez que se mantiene un cierto grado de cooperación entre procesadores.

- El llamado modelo de *difusión* (o single-population fine-grained) [CAN97] trabaja con una población de tamaño muy reducido, que puede oscilar entre un solo individuo o unos pocos. La diferencia con respecto al modelo de islas radica en que el esquema de difusión requiere una estructura topológica de los procesadores que lleve a cabo selección y recombinación de forma eficiente.

En el caso de paralelizar meta-heurísticas multi-objetivo basadas en búsqueda local (MOLSAs) [RAN99, CUN01, CAH04], los modelos de paralelización incluyen otras aproximaciones [CUN01]:

- El modelo de *vecindario paralelo* divide el vecindario en reparticiones que se exploran de forma simultánea. Esto resulta particularmente interesante cuando la evaluación de las soluciones es muy costosa y/o el tamaño del vecindario es grande.

- El modelo *multi-arranque* consiste en ejecutar en paralelo varios buscadores locales sin intercambio de información. Este modelo intenta mejorar la calidad de la búsqueda aprovechando la diversidad ofrecida por cada ejecución independiente, y haciendo uso de diferentes parámetros.

6.3. PrMSATS: Paralelización de rMSATS

Como comentamos anteriormente, el procesamiento paralelo es recomendable dado el carácter NP-completo del problema de repartición de grafos, ya que el tamaño de los grafos de prueba puede llegar a ser muy elevado. Diversos autores han diseñado algoritmos paralelos para resolver el problema de repartición de grafos [DIE96, WAL00, SCH02].

6.3.1. Descripción de la Paralelización

En el Capítulo 4 se describió el funcionamiento del procedimiento rMSATS [GIL02]. A continuación se presenta PrMSATS [BAÑ04a], que analiza las ventajas de explorar el espacio de búsqueda utilizando diferentes particiones iniciales y esquemas de enfriamiento considerando el uso del procesamiento paralelo. Mientras que rMSATS utilizaba una sola solución durante el proceso de búsqueda, PrMSATS trabaja con una población de soluciones, de forma que cada procesador trabaja con una o varias soluciones de forma simultánea. En concreto se aplica el modelo de islas, de forma que si, por ejemplo, se trabaja con una población de P_{tam}=20 individuos, y se dispone de 4 procesadores (islas), cada uno de ellos procesará 5 individuos de forma independiente. Aunque la lectura del grafo desde el archivo es idéntica para todos los individuos, esta se realiza de forma simultánea en cada procesador, evitando el tiempo de comunicaciones que se requeriría en el caso de hacerlo en un solo procesador, para posteriormente migrarlo al resto.

Cada procesador aplica FGA [FAH88] en el grafo objetivo partiendo de un vértice aleatorio elegido mediante la estrategia de selección aleatoria por intervalos [BAÑ04b]. Esta estrategia consiste en que, dada una población de P_{tam} individuos, el individuo P_i (i=1..P_{tam}) realiza la primera partición desde un vértice aleatorio perteneciente al intervalo $\left[\frac{i-1}{P_{tam}} \cdot |V|, \frac{i}{P_{tam}} \cdot |V|\right[$, de forma que se asegura la diversidad en las particiones iniciales. Una vez que cada procesador ha obtenido las particiones iniciales de sus soluciones asignadas, el siguiente paso es aplicar la fase de refinamiento haciendo uso de rMSATS. Sin embargo, antes de llevar a cabo este proceso de optimización, cada procesador asigna para cada una de las soluciones gestionadas unos parámetros de enfriamiento determinados. Para ello, los valores de temperatura iniciales para cada solución difieren en función de su identificador i, como detallamos a continuación: primeramente se establece un intervalo de temperaturas iniciales $[Ti_{min}, Ti_{max}]$. La solución P_1 parte de Ti_{max}, la solución P_{tam} parte de Ti_{min}, y el resto se distribuyen en dicho intervalo a la misma distancia. En la Figura 6.2 se muestra un ejemplo de dicha estrategia, donde se ha establecido un intervalo de temperaturas iniciales, T_i=[150,50], y un número fijo de iteraciones, n_i=1000. El individuo A tiene el valor más elevado de T_i, por lo que T_{enfr} es bajo, y la temperatura decrece rápidamente. Con el individuo J, sucede lo contrario, ya que parte de un valor inferior de T_i, y un T_{enfr} superior. Esta estrategia asegura un equilibrio en el número de iteraciones de optimización a realizar por cada individuo.

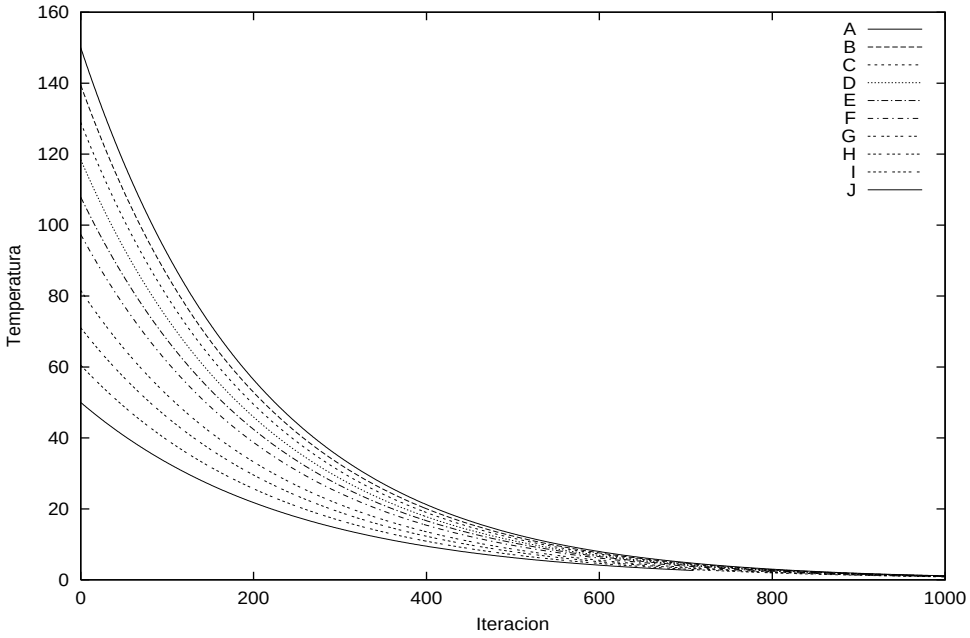

Figura 6.2: Alternativas de enfriamiento.

PrMSATS también incluye aspectos cooperativos en la búsqueda, ya que se permiten migraciones esporádicas de individuos pertenecientes a diferentes procesadores (islas) con el objetivo de añadir un cierto grado de elitismo. En concreto la fase de partición inicial y asignación de parámetros de enfriamiento se hace de forma independiente. Es en la fase de refinamiento en la cual, pese a que cada procesador trabaja de forma independiente con sus individuos, se permite de forma esporádica migrar los mejores individuos entre procesadores. La selección de los individuos a migrar se realiza mediante un torneo binario elitista [GOL91]. Dichas migraciones se gestionan haciendo uso de diferentes políticas. En concreto se han utilizado tres estrategias diferentes. La primera política, Figura 6.3(a), está basada en comunicar solamente los procesos vecinos, esto es, el proceso P_i sólo puede intercambiar individuos con P_{i-1} y P_{i+1} de forma alternativa. Esta política corresponde a una implementación en grano fino. La segunda política de migración, Figura 6.3(b), está basada en una migración incremental, esto es, el número de procesadores que intercambian información se incrementa en función del número de comunicaciones previamente realizadas. Esta idea se basa en que al inicializarse la búsqueda, el valor de t es alto, razón por la que las migraciones deben reducirse a un cierto número de individuos, mientras que

en las siguientes iteraciones el valor de t disminuye, y con ello la probabilidad de aceptar movimientos que empeoren la calidad de las particiones, razón por la cual migran los mejores individuos de la población a un mayor número de procesadores. La tercera política de migración, Figura 6.3(c), está basada en enviar los mejores individuos de la población a todos los procesadores (broadcast), con lo que el grado de elitismo es máximo. Un factor de gran importancia en este proceso migratorio es el hecho de establecer una frecuencia de migración adecuada, incluso más en problemas tan complejos como el de repartición de grafos, donde la topología y tamaño del grafo resulta determinante. Considerando las características de rMSATS e intentando evitar la convergencia prematura de las soluciones, las comunicaciones han sido establecidas como detallamos a continuación: dado NC el número de comunicaciones, y n_i el número de iteraciones, ambos previamente establecidos, las comunicaciones entre procesos utilizando alguna de las políticas de migración previamente descritas, se llevan a cabo en las iteraciones $\{\frac{n_i}{NC}, 2 \cdot \frac{n_i}{NC}, ..., NC \cdot \frac{n_i}{NC}\}$. En nuestros experimentos, hemos considerado un valor de NC=5. Al finalizar la fase de refinamiento todos los procesadores envían información sobre el valor de aptitud de sus individuos a uno de los procesadores, que retorna el mejor de los individuos obtenidos.

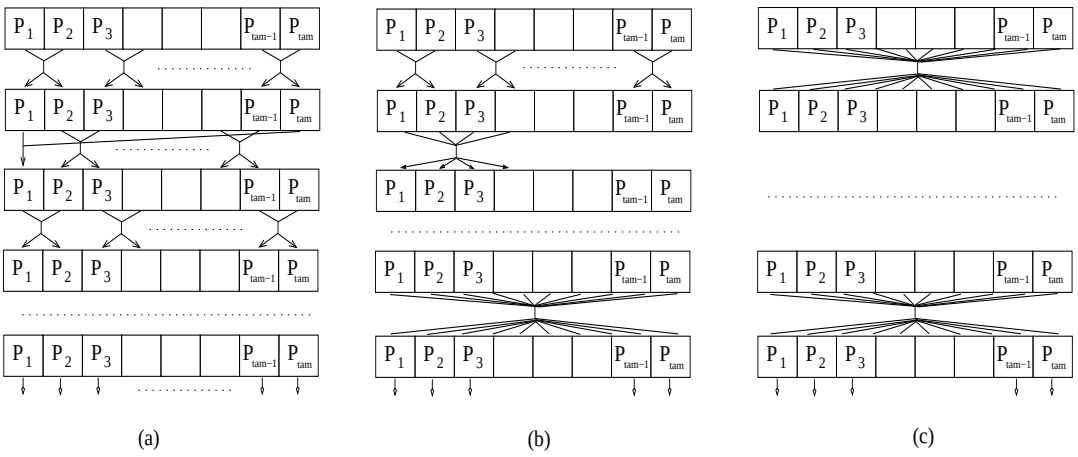

Figura 6.3: Políticas de migración implementadas.

El Algoritmo 6.3.1 presenta formalmente el procedimiento PrMSATS. En el primer paso se asignan los parámetros de enfriamiento a utilizar en el proceso de optimización, para aplicar posteriormente FGA y obtener la partición inicial. En el siguiente paso se inicializan los parámetros de enfriamiento con los datos

calculados anteriormente, para aplicar de nuevo el bucle de optimización en el nivel actual, controlando el número de iteraciones del proceso de refinamiento. En cada ciclo de este bucle se aplica rMSATS con sus propios parámetros. Periódicamente se permite la comunicación entre procesadores utilizando alguna de las políticas de migración descritas anteriormente. Tras finalizar el proceso de búsqueda se envían todas las soluciones al procesador maestro, el cual devuelve la mejor de todas las soluciones recibidas.

Algoritmo 6.3.1

Begin PrMSATS ($G, SG, DB_{max}, \mu, Ti_{min}, Ti_{max}, T_{enfr}, T_{parada}, polit\_migr, mR$)
  Desde `Conta_P=1` **hasta** `Conta_P=`P_{tam}
    `Calcular` T_i `y` T_{enfr} `para este individuo;`
    $t \leftarrow T_i$;
    $G \leftarrow$`Obtener partición inicial mediante FGA;`
  Hacer
    `Aplicar iteración rMSATS();`
    `iterac_actual++;` $t \leftarrow t \cdot T_{enfr}$;
    Si (`iterac_actual` **MOD** mR)**=0** **entonces**
      `Migración(`*polit_migr*`);`
  Mientras ($t > T_{parada}$);
  `Enviar la mejor solución obtenida a un procesador central;`
  `El procesador central retorna la mejor solución recopilada;`
End PrMSATS

6.3.2. Análisis Experimental

La Tabla A.8 muestra los resultados obtenidos por PrMSATS al utilizar 1, 5 y 20 individuos con particiones iniciales diferentes. Todas ellas se han calculado mediante FGA partiendo de diferentes vértices iniciales, utilizando T_i=100 y T_{enfr}=0.995 como valores fijos. Como podemos ver, al incrementar el número de soluciones diferentes de la población inicial se consigue un mayor grado de diversidad, que permite mejorar la calidad de las soluciones en la mayoría de los casos, en comparación con la ejecución de rMSATS (PrMSATS[1]).

En la Figura 6.4 se muestran los resultados obtenidos al utilizar diferentes valores de T_i, haciendo uso de poblaciones de soluciones de tamaño P_{tam}=20. Los resultados indican que el uso de diferentes valores de T_i y T_{enfr} obtiene mejores resultados que cuando se utilizan parámetros fijos (T_i=100 y T_{enfr}=0.995). La modificación del rango de temperaturas iniciales no tiene un efecto homogéneo, ya que ninguno de los tres intervalos seleccionados obtiene una clara mejora en comparación con los otros, aunque si sobre el caso de utilizar parámetros de temperatura fijos.

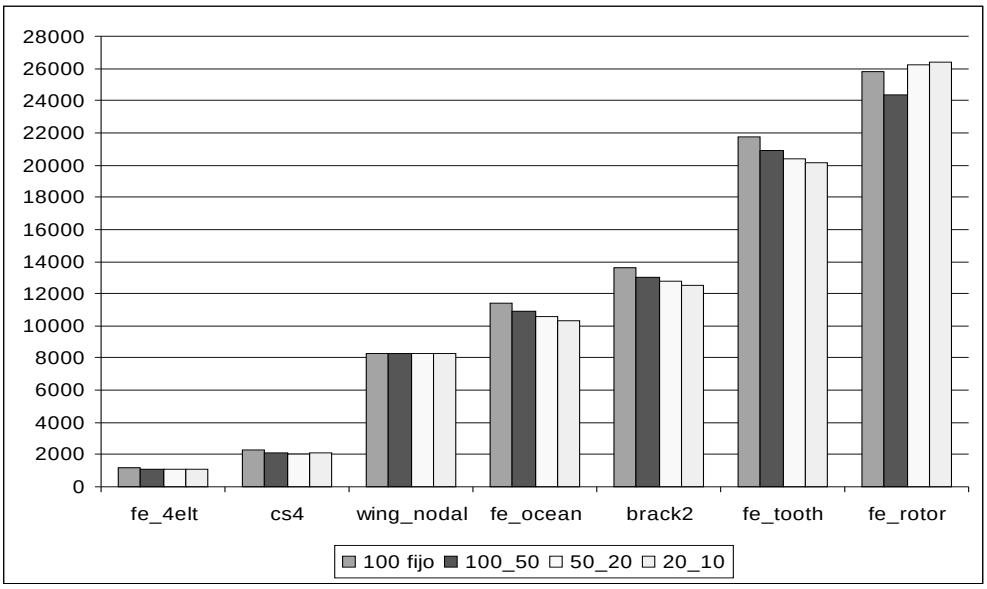

Figura 6.4: Efecto de utilizar diferentes temperaturas iniciales.

En la Tabla A.9 se detallan los resultados obtenidos por los tres esquemas de migración implementados en PrMSATS. Sobre estos resultados se puede determinar que la política de migración binaria de grano fino (PrMSATS^A) obtiene el mejor resultado en el 11 % de los casos, la incremental (PrMSATS^B) en el 31 %, y la de broadcast (PrMSATS^C) en el 35 %, mientras que en el resto de casos, 23 %, al menos dos de las versiones obtienen la misma solución.

Aunque el objetivo de la paralelización es optimizar las soluciones, también se ha analizado la ganancia de velocidad obtenida (speedup). La Figura 6.5(a) muestra el speedup obtenido por algunos de los grafos de prueba utilizados, considerando que SG=4. Este speedup se ha calculado considerando el tiempo requerido por el algoritmo paralelo en encontrar la misma solución que la versión

secuencial, considerando solamente el tiempo de procesamiento tras realizar las particiones iniciales. El número de procesadores utilizado para dicho análisis ha sido np={1,2,4,8,16}. Como podemos observar, pese a que el paralelismo se ha enfocado a mejorar la exploración del espacio de búsqueda, el speedup obtenido muestra que el algoritmo paralelo encuentra la misma solución obtenida por la versión secuencial de forma más rápida, aunque la eficiencia del algoritmo paralelo disminuye conforme aumenta el número de procesadores (ver Figura 6.5(a)).

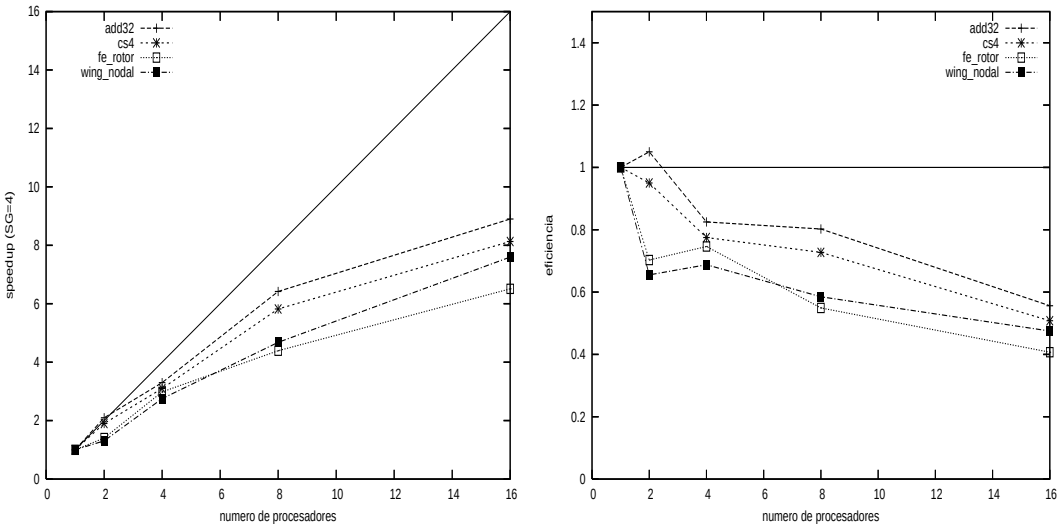

Figura 6.5: Speedup y eficiencia en PrMSATS para SG=4 variando el número de procesadores np={1,2,4,8,16} en varios grafos de prueba.

6.4. PMSATS: Paralelización de MLrMSATS

Con anterioridad, algunos autores han propuesto paralelizaciones de implementaciones multi-nivel [WAL00]. En esta sección presentamos la versión paralela de MLrMSATS, aquí denominada PMSATS (Parallel Multilevel Simulated Annealing and Tabu Search) [BAÑ04b].

6.4.1. Descripción de la Paralelización

PMSATS utiliza un esquema de paralelización muy similar al de PrMSATS. En ambos casos cada procesador trabaja con un conjunto de soluciones inde-

pendientes que evolucionan utilizando rMSATS, y que esporádicamente pueden migrar entre procesadores para proseguir la búsqueda haciendo uso de las soluciones más prometedoras encontradas hasta el momento. Sin embargo, existen algunas diferencias dado el carácter multi-nivel de PMSATS. En concreto cada procesador aplica la estrategia de agrupamiento HEM partiendo de un vértice independiente obtenido por selección aleatoria por intervalos, descrita anteriormente. Del mismo modo, cada procesador calcula las particiones iniciales mediante el procedimiento FGA [FAH88]. Tras esto, cada procesador lleva a cabo la fase de refinamiento de forma independiente, aunque también se permiten migraciones. Al igual que en PrMSATS, el criterio de selección utilizado ha sido torneo binario elitista [GOL91], es decir, que el ganador del torneo envía su solución a los demás, los cuales continuarán aplicando rMSATS a partir de dicha solución utilizando sus propios valores de t y T_{enfr}. De esta forma, el algoritmo continua explorando las mejores soluciones utilizando diferentes parámetros de SA. En este caso se han analizado dos de las políticas de migración uilizadas en PrMSATS. En concreto se ha implementado la política de migración binaria incremental (Figura 6.3(a)) y la polítuca de broadcast (Figura 6.3(c)). La elección de ambas polĺiticas radica en que presentan grados de elitismo opuestos.

Al igual que en PrMSATS, cada procesador realiza la partición inicial de los individuos que controla aplicando FGA desde un vértice que depende de su identificador de proceso. La fase de refinamiento también se ejecuta independientemente, aunque al igual que en PrMSATS se permiten migraciones esporádicas de individuos de la población. Es aquí donde entran en juego las diferentes políticas y frecuencias de migración propuestas. Al finalizar la fase de refinamiento todos los procesadores envían información de sus individuos a un procesador central, el cual determina la mejor de todas las soluciones obtenidas.

Un aspecto diferenciador de PMSATS con respecto a PrMSATS es la estrategia utilizada para seleccionar las iteraciones en las que se realiza el proceso de cooperación (migraciones). Mientras que en PrMSATS, al ser una paralelización no multi-nivel, sólo es posible migrar individuos en el grafo original, en el caso de PMSATS se puede plantear la cuestión de si es más conveniente migrar individuos sólo en el nivel superior, como en PrMSATS, o también en los niveles intermedios. En concreto se han implementado dos estrategias de migración diferentes. La primera (M1) consiste en migrar los individuos tras cada paso de refinamiento. Así, tras aplicar rMSATS en el nivel actual, y antes de subir de nivel, los individuos se seleccionan utilizando alguna de las políticas de migración que describimos anteriormente. En la segunda alternativa (M2) las

comunicaciones se han establecido solamente en el nivel superior, tal y como se detalló para PrMSATS.

Algoritmo 6.4.1

Begin PMSATS (G,SG,DB_{max},μ,Ti_{min},Ti_{max},T_{enfr},T_{parada}, *polit_migr*, *frec_migr*, mR, *estr_agrup*)

Desde `Conta_P=1` **hasta** `Conta_P=`P_{tam}

$NN \leftarrow$`Calcular_Número_Niveles_Agrupamiento(`SG,μ`)`

`Agrupar_Grafo(`G,NN,`estr_agrup);`

$G \leftarrow$`Obtener partición inicial mediante FGA;`

Desde `NivelActual=NN hasta NivelActual=1` **hacer**

`Calcular` T_i `y` T_{enf} `para este individuo;`

$t \leftarrow T_i$;

Hacer

`Aplicar iteración rMSATS();`

`iterac_actual++;` $t \leftarrow t \cdot T_{enfr}$;

Si `((frec_migr=M2) & (NivelActual=1))`

`& (iterac_actual` **MOD** mR`)=0` **entonces**

`Migración(`*polit_migr*`);`

Mientras ($t > T_{parada}$);

`Desagrupar` G`[`*NivelActual*`]` `un nivel;`

Si `(frec_migr=M2)` **entonces**

`Selección_Elitista(`*polit_migr*`);`

`Enviar la mejor solución obtenida a un procesador central;`

`El procesador central retorna la mejor solución recopilada;`

End PMSATS

A modo de resumen, la Figura 6.6 muestra las diferencias entre rMSATS, MLrMSATS y PMSATS. La Figura 6.6(a) corresponde a rMSATS, donde sólo se optimiza una solución tras obtener la partición inicial con el procedimiento FGA. La Figura 6.6(b) corresponde al funcionamiento de MLrMSATS donde de nuevo se optimiza una sola solución pero haciendo uso del paradigma multinivel. Finalmente, la Figura 6.6(c) corresponde a PMSATS, que trabaja con una población de soluciones haciendo uso de procesamiento paralelo. Las fases

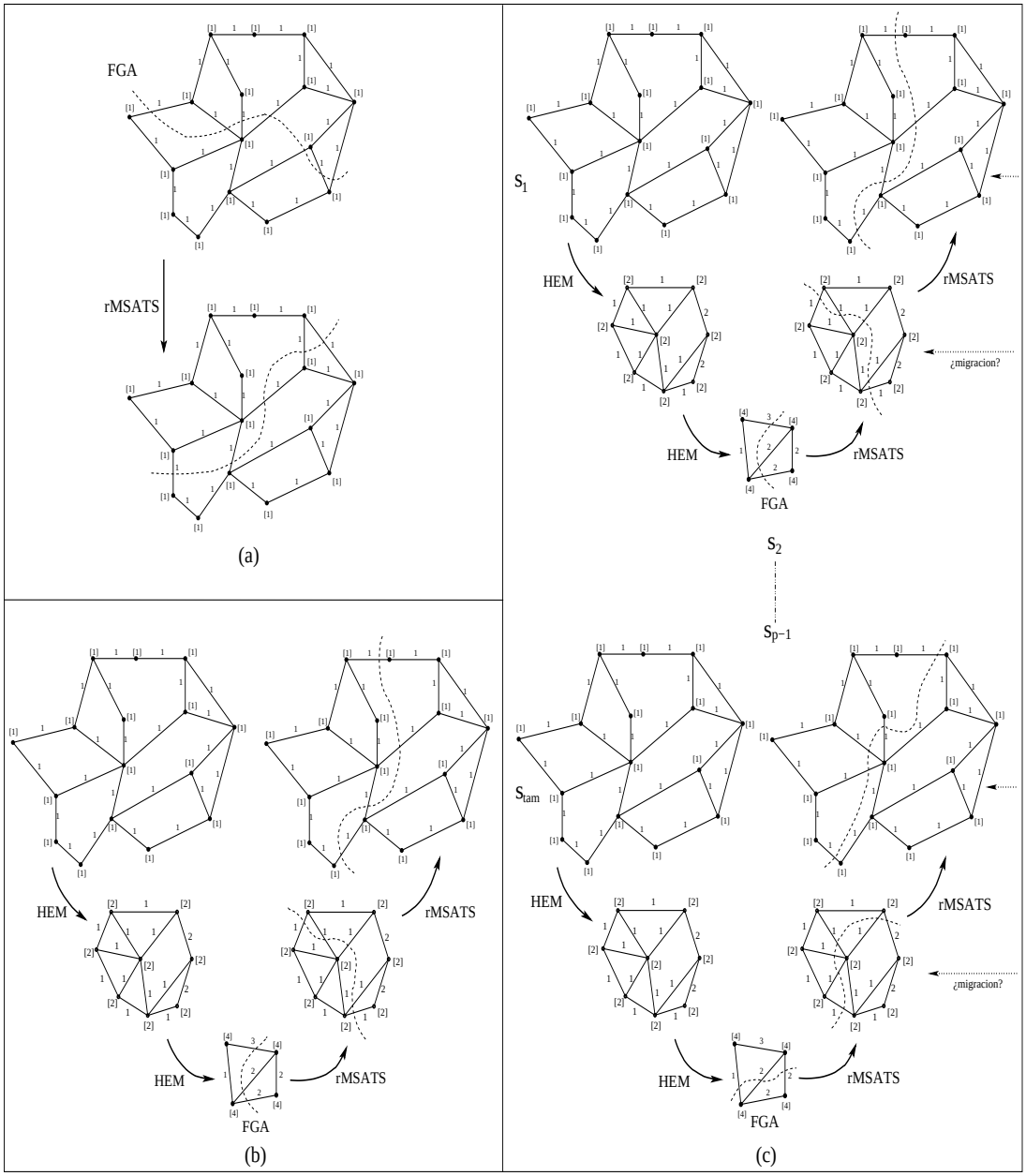

Figura 6.6: Descripción de rMSATS, MLrMSATS y PMSATS.

de agrupamiento y repartición inicial se realizan de forma independiente en cada procesador. La fase de refinamiento se lleva a cabo mediante la aplicación de rMSATS con los valores de enfriamiento propios, de forma que en ciertas iteraciones existen comunicaciones entre procesadores.

El Algoritmo 6.4.1 presenta formalmente el procedimiento PMSATS. En el primer paso, se adquieren los parámetros de entrada. Posteriormente cada solución s correspondiente al grafo G se agrupa un total de NN niveles, donde NN depende de μ y SG, utilizando HEM. Tras esto, se calculan los parámetros de enfriamiento a utilizar en el proceso de optimización, para pasar a aplicar posteriormente FGA y obtener la partición inicial. Se inicializan los parámetros de enfriamiento con los valores calculados anteriormente, para aplicar de nuevo el bucle de optimización en el nivel actual, controlando el número de iteraciones del proceso de refinamiento. En cada ciclo de este bucle se aplica rMSATS con sus propios parámetros que son actualizados continuamente. Periódicamente, se permite la comunicación entre procesadores haciendo uso de la política y frecuencia de migración establecida. Cuando la temperatura cae por debajo de la de parada, el grafo se desagrupa, y se sube de nivel, tomando de nuevo los valores de enfriamiento originales. Al final del proceso, todos los procesadores envian sus soluciones al procesador maestro, el cual devuelve la mejor de todas las soluciones recibidas.

6.4.2. Análisis Experimental

La Figura 6.7 compara la aplicación de PMSATS con las siguientes configuraciones, (P_{tam}=10, n_i=3000), (P_{tam}=20, n_i=1500) y (P_{tam}=40, n_i=750). Recordamos que el valor 1 en el eje de ordenadas representa el número de cortes obtenido por la mejor configuración comparada, y que actuará como referencia para calcular el incremento del número de cortes obtenido en el resto de configuraciones para dicho grafo. Aquí, los valores de enfriamiento son T_i=100 y T_{enfr}=0.995. Como se puede observar en dicha figura, la mejor configuración es P_{tam}=20 y n_i=1500. La ventaja de utilizar un mayor número de individuos (P_{tam}=40, n_i=750) radica en disponer de mayor diversidad en el proceso de búsqueda, aunque debido a la complejidad de la misma es necesario aplicar rMSATS durante un número de iteraciones superior a n_i=750. Por el contrario utilizar pocos individuos (P_{tam}=10, n_i=3000) resulta menos eficiente a la hora de explorar el espacio de búsqueda.

Otro aspecto analizado es determinar qué rango de temperaturas se esco-

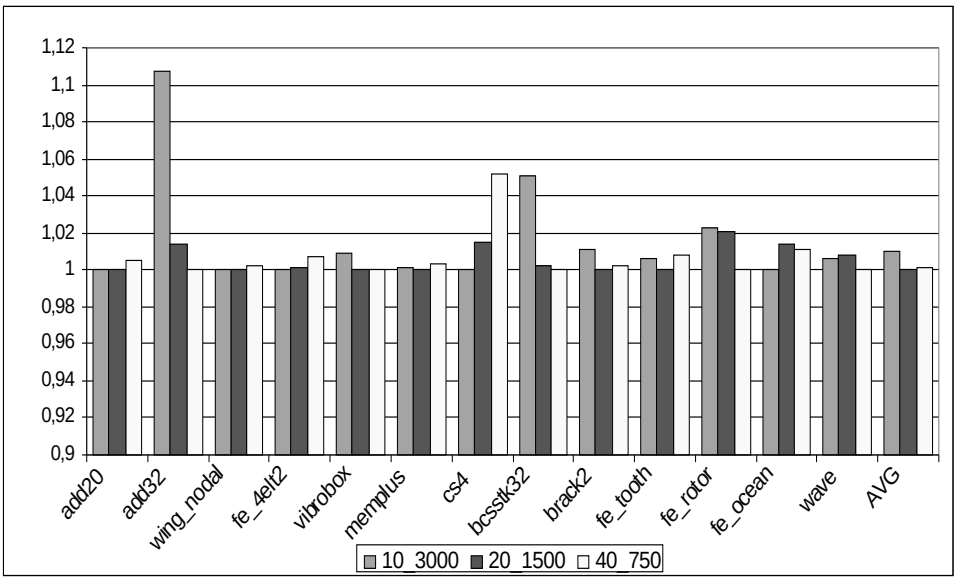

Figura 6.7: Efecto de variar los individuos y las iteraciones.

ge durante el proceso de enfriamiento. Aunque con valores como T_i=100 y T_{enfr}=0.995 se obtienen resultados aceptables, no se puede concluir que sean los valores óptimos. Con el objetivo de analizar este factor, planteamos aplicar una estrategia de división recursiva por rangos. La idea consiste en seleccionar un rango de temperaturas iniciales extenso, por ejemplo T_i*=[500,2]. Este intervalo se divide recursivamente en dos partes iguales hasta que ninguno de los sub-intervalos de un cierto nivel mejore las soluciones obtenidas por el intervalo de temperaturas superior, el cual será seleccionado como intervalo adecuado. La Figura 6.8 muestra el número medio de cortes obtenido por cada intervalo tras ejecutar el algoritmo sobre esos grafos de prueba para SG={4,16,64}. Como se observa, para un tamaño de población P_{tam}=20, ninguno de los cuatro sub-intervalos es capaz de conseguir, en media, un número medio de cortes inferior al obtenido por T_i*=[500,250], por lo que este será el intervalo utilizado en experimentos posteriores.

En la Figura 6.9 se muestra el rendimiento del algoritmo para las diferentes políticas y frecuencias de migración. En esas ejecuciones, se ha utilizado una población de P_{tam}=20 individuos y n_i=1500 iteraciones, utilizando el rango de temperaturas Ti*=[500,250]. Los resultados obtenidos indican que la implementación binaria de grano fino mejora el número medio de cortes con respecto a

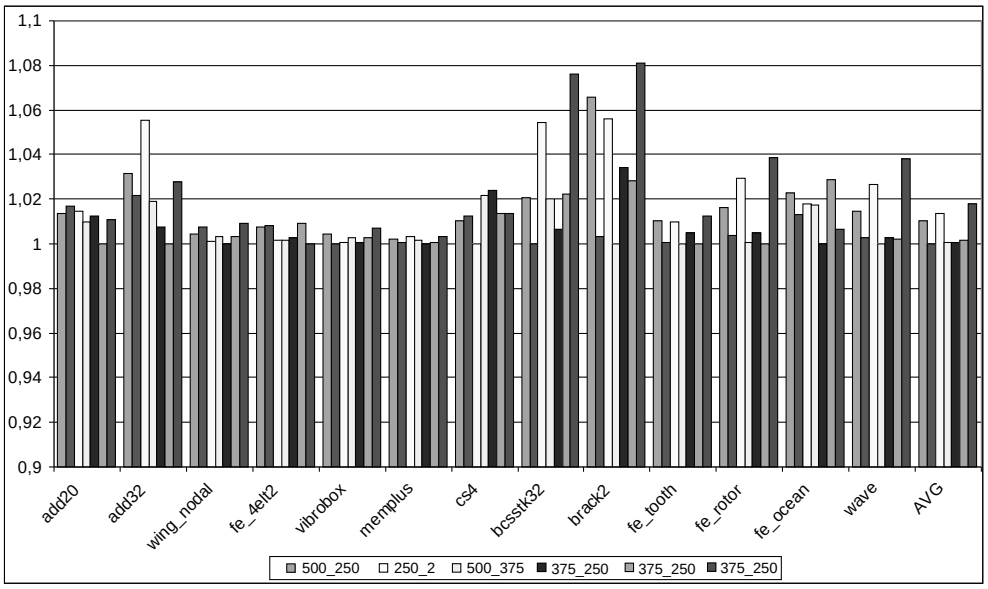

Figura 6.8: Uso de diferentes rangos de temperatura.

utilizar la estrategia de broadcast. La razón es que si se introduce un elevado elitismo (broadcast) se rompe la independencia en la búsqueda realizada por cada individuo. Con respecto a la frecuencia de migración, los resultados indican que es mejor utilizar comunicación solamente en el nivel superior (M2) que migrar en cada cambio de nivel (M1). La razón de este comportamiento es que resulta conveniente dejar un cierto grafo de libertad a cada individuo en los niveles intermedios, con la idea de mantener la diversidad en la búsqueda, y aplicar el elitismo solamente en el nivel superior.

Las Tablas A.10 y A.11 comparan los mejores resultados obtenidos por PMSATS frente a ParMETIS [MES], y frente a las mejores soluciones conocidas incluidas en el archivo de partición de grafos [GPA], sobre algunos de los grafos de prueba descritos en el Capítulo 3, con $DB_{max} \leq 5\,\%$. En comparación con ParMETIS, PMSATS obtiene mejores resultados en el 95 % de los casos, mientras que ParMETIS sólo obtiene la mejor partición en el 1 % de todas las ejecuciones. En el resto de casos, 4 %, PMSATS y ParMETIS obtienen el mismo número de cortes. Por otro lado, PMSATS obtiene mejores resultados que las mejores soluciones conocidas [GPA] en el 40 % de las ejecuciones, e igual a la mejor solución en el 12 % de casos. Los tiempos de ejecución de PMSATS pueden oscilar entre unos pocos segundos y algunas horas, dependiendo del grafo a reparticionar.

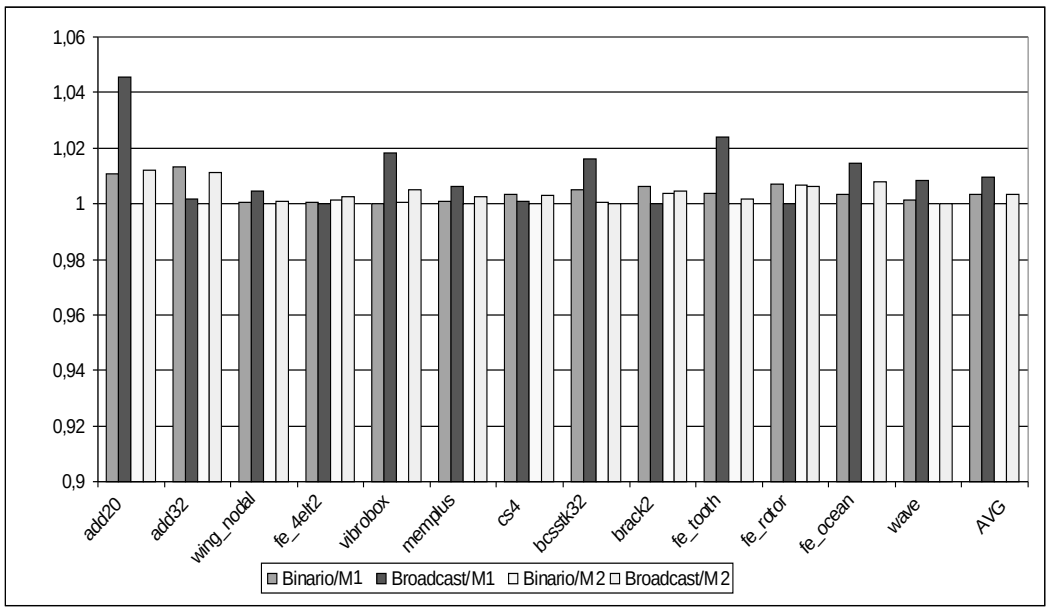

Figura 6.9: Resultados obtenidos tras aplicar las políticas de migración.

Aunque los tiempos de ejecución de PMSATS son superiores a los de ParMETIS, son inferiores a muchos de los algoritmos incluidos en [GPA].

Al igual que para PrMSATS, vamos a analizar la ganancia de velocidad obtenida (speedup) considerando el tiempo requerido por el algoritmo paralelo en encontrar la misma solución que la versión secuencial, pero en este caso considerando las tres fases del algoritmo multi-nivel. El número de procesadores utilizado para dicho análisis ha sido de np=$\{1,2,4,8,16\}$. De los resultados obtenidos (ver Figura 6.10) obtenemos conclusiones similares a las obtenidas para PrMSATS, esto es, pese a que el paralelismo ha sido enfocado a explorar diferentes áreas del espacio de búsqueda, el speedup obtenido muestra que el algoritmo paralelo encuentra la misma solución obtenida por la versión secuencial de forma más rápida. No obstante conforme aumenta el número de procesadores se observa un deterioro en la eficiencia, que como se puede observar es más acusada que en PrMSATS. Esto es debido a que la inmensa mayoría de las mejores soluciones se obtienen en el nivel superior (grafo original), por lo que las fases de agrupamiento, reparticionamiento inicial, y refinamiento en PMSATS actuan como un procedimiento secuencial, y por tanto el deterioro en el speedup es mucho mayor que en el caso de PrMSATS.

La Figura 6.11 muestra las mejores soluciones conocidas obtenidas por ca-

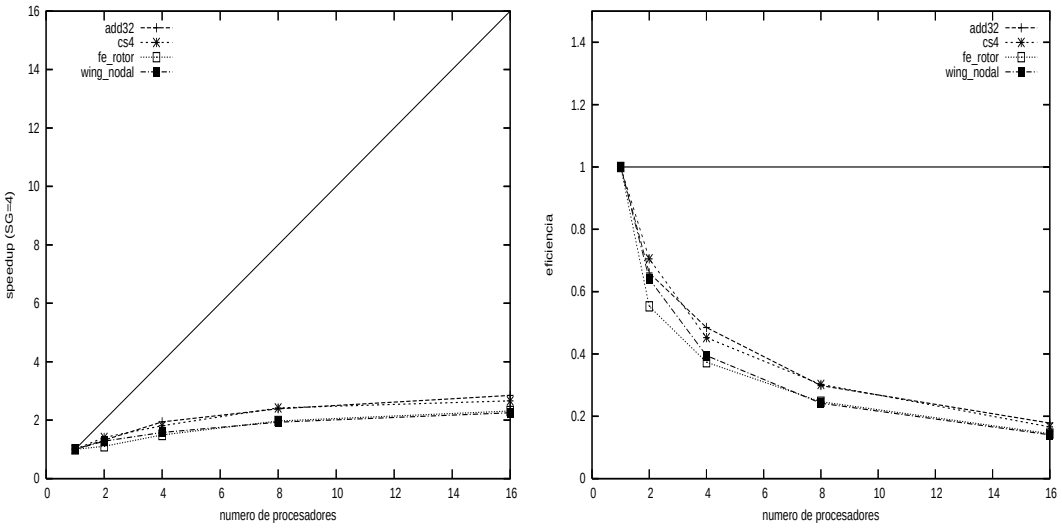

Figura 6.10: Speedup y eficiencia en PMSATS para SG=4 variando el número de procesadores np={1,2,4,8,16} en varios grafos de prueba.

da uno de los algoritmos incluidos en [GPA] para SG={2,4,8,16,32,64}. Si dos o más algoritmos diferentes obtienen el mismo número de cortes, se considera como mejor solución aquella que tenga un desbalanceo menor. Si además del número de cortes, el desbalanceo obtenido es el mismo, entonces se considera como mejor algoritmo aquel que la encontró con anterioridad. Como se muestra en la Figura 6.11, PMSATS es claramente el mejor algoritmo, considerando el número de soluciones óptimas obtenidas. PMSATS también obtiene, en algunos casos, la mejor partición encontrada por otros algoritmos previamente propuestos. Tanto JOSTLE Evolutionary (JE) [SOP04] como iterated JOSTLE (iJ) [WAL04] consiguen también un número importante de mejores soluciones conocidas, aunque con tiempos de ejecución muy elevados (algunas ejecuciones de JE requieren tiempos de ejecución de varios días [SOP04]). El resto de los algoritmos obtienen la mejor solución en un número de casos inferior.

6.5. pPSA: Paralelización de Pareto Simulated Annealing

Con el objetivo de analizar diferentes técnicas de paralelización haciendo uso de un método encontrado en la literatura, se plantea la paralelización de PSA.

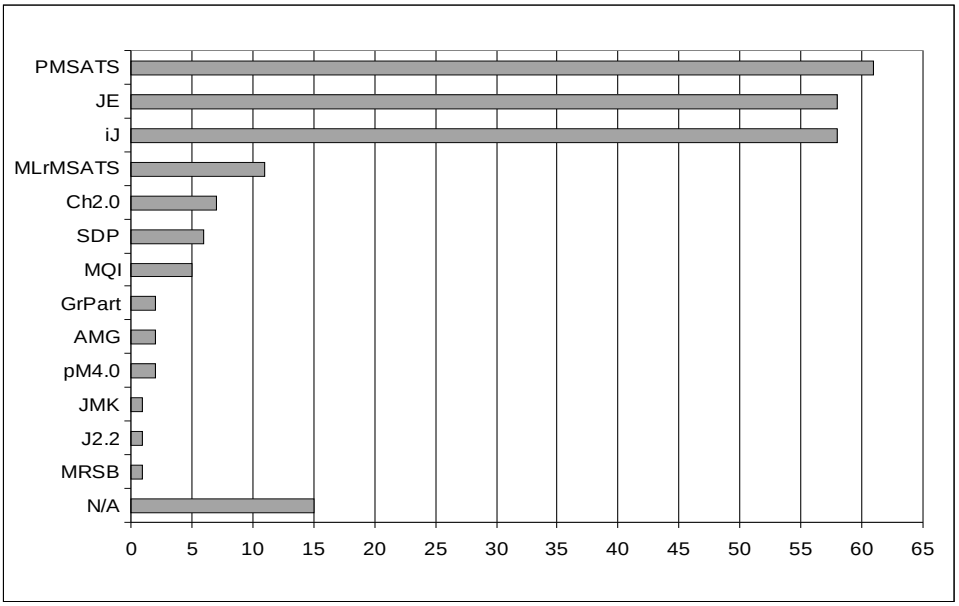

Figura 6.11: Ranking de algoritmos por número de mejores resultados conocidos.

Como se describió en el Capítulo 2, Pareto Simulated Annealing [CZY98] es una MOMH poblacional basada en enfriamiento simulado. Dicha paralelización, la primera que se realiza sobre PSA, amplía las realizadas anteriormente. Mientras que en PrMSATS y PMSATS se paralelizaba mediante el modelo de islas con el objetivo de mejorar la calidad de las soluciones, aquí tomamos como base un algoritmo previamente propuesto y analizamos, no sólo el modelo de islas, sino además otros paradigmas de paralelización alternativos. En concreto hemos implementado cuatro versiones paralelas de Pareto Simulated Annealing (PSA) [CZY98]: multi-arranque, maestro-esclavo, islas, e islas con división del espacio objetivo. A continuación se describen las diferentes estrategias de paralelización incluidas en pPSA (parallel Pareto Simulated Annealing) [BAÑ06a], y cuyas características son resumidas en la Tabla 6.1.

6.5.1. Descripción de la Paralelización

- *Versión paralela multi-arranque de PSA (*MS*).* A la hora de paralelizar cualquier algoritmo es habitual preguntarse si resulta más eficiente ejecutar una sola ejecución durante n_i iteraciones (o n_e evaluaciones) o ejecutar NE ejecuciones heterogéneas durante n_i/NE iteraciones (evaluaciones).

Para contestar esta cuestión en la paralelización de PSA se presenta una implementación paralela multi-arranque (*MS*), que consiste en ejecutar *np* ejecuciones independientes de PSA durante n_i/np iteraciones cada una.

- *Versión paralela maestro-esclavo de PSA (*MW*).* Como detallamos anteriormente la versión maestro-esclavo mantiene la secuencialidad de los algoritmos. Una de las características principales de PSA [CZY98] es el uso de un parámetro que controla la importancia de cada objetivo, el cual es variado en función de las relaciones de dominancia entre individuos. Dada esta circunstancia, la única paralelización que preserva la secuencialidad del algoritmo es la búsqueda de soluciones vecinas. Así pues, el procesador maestro crea y distribuye la población inicial al resto de procesadores. Cada procesador, incluido el maestro, evalua P_{tam}/np soluciones, tras lo cual cada procesador envia sus individuos al procesador maestro, el cual se encarga de evaluarlos como en PSA. Este proceso se repite hasta que se cumple la condición de parada, devolviendo las soluciones localizadas en el archivo externo de soluciones no dominadas (*ND*).

- *Versión paralela de islas de PSA (*I*)*: existen aplicaciones en las que las comunicaciones entre procesadores son tan costosas que el esquema maestro-esclavo se convierte en un mecanismo muy lento. Una forma de reducir el tiempo de ejecución es hacer uso del modelo basado en islas (*I*). El funcionamiento consiste en ejecutar PSA de forma independiente en la sub-población asignada. Sin embargo, de forma periódica los procesadores cooperan mediante el intercambio de ciertos individuos encontrados en cada isla con la idea de continuar con el proceso de búsqueda por las áreas más prometedoras. La selección adecuada del valor *mR* se convierte en otro problema de optimización, por lo que se ha comparado el rendimiento obtenido utilizando varios valores de *mR*.

- *Versión paralela basada en el modelo basado en islas con división del espacio objetivo de PSA (*I_{deo}*).* Recientemente, Toro y col. [TOR04] aplicaron una variante del modelo de islas consistente en dividir la búsqueda de forma que cada procesador trabaje en un área diferente del espacio objetivo. De esta forma, la población entera se divide en *np* sub-poblaciones y cada sub-población trabaja en una región particular. Para conseguir que cada procesador trabaje en un área determinada del espacio objetivo es necesario descartar aquellas soluciones del espacio de búsqueda cuya traslación al

espacio objetivo estén fuera de su ámbito. Así, para el caso del problema de repartición de grafos cada procesador va a trabajar en un rango determinado de desbalanceo, de forma que se van a descartar aquellas soluciones cuyo desequilibrio quede fuera de su rango asignado.

Tabla 6.1: Implementaciones paralelas: principales características.

versión	NE	$\frac{n_i}{ejecución}$	$\sum n_i$	$\frac{sub-poblaciones}{NE}$	|sub-población|	mR
PSA	1	R	R	1	P_{tam}	–
MS	r	$\frac{R}{r}$	R	1	P_{tam}	–
MW	1	R	R	np	$\frac{Ptam}{np}$	1
I	1	R	R	np	$\frac{Ptam}{np}$	mR
I_{deo}	1	R	R	np	$\frac{Ptam}{np}$	mR

6.5.2. Análisis Experimental

Como en el resto de algoritmos basados en partición de grafos, las soluciones iniciales se obtienen por el procedimiento FGA mediante la selección por intervalos aleatorios propuesta en [BAÑ04a] y descrita en el Capítulo 4. El proceso de búsqueda local se lleva a cabo utilizando el operador que mueve vértices entre sub-grafos vecinos. Con el objetivo de evaluar el impacto del tamaño de la población hemos utilizado dos configuraciones diferentes: P_{tam}=$\{32,128\}$. Como PSA también elimina las soluciones no dominadas del archivo externo (ND), este no ha sido limitado en tamaño. Dado que determinar los parámetros de enfriamiento adecuados es una tarea difícil, los valores típicos son T_i=100, T_{enfr}=0.99. Considerando que el proceso de optimización finaliza cuando t<T_{parada}=0.01, esos valores implican que el número de iteraciones es n_i=920. Sin embargo, con el objetivo de reducir el efecto de seleccionar diferentes parámetros de enfriamiento se ha utilizado una planificación independiente en cada solución de la población. En particular, se ha utilizado la estrategia propuesta en [BAÑ04b], y descrita en la Figura 6.2. A la hora de analizar el análisis de la ganancia de velocidad, se ha utilizado np=$\{1,2,4,8,16\}$ procesadores. Con la intención de evaluar el impacto de usar diferentes ratios de migración en los modelos de islas, se utilizan los siguientes valores mR=$\{5,20,100,920\}$ (mR=1 sería similar al modelo MW debido a que cada isla se comunica con los otros en todas las iteraciones, mientras que con mR=920 se lleva a cabo migración solamente cuando se finaliza el proceso de búsqueda). Cuando mR se incrementa la independencia de

cada isla se incrementa, mientras que el número de comunicaciones (y tiempo de ejecución) disminuye. Los resultados corresponden a la repartición de algunos de los grafos de prueba en SG=16 sub-grafos.

Los resultados de este análisis se dividen en dos partes. Primeramente se analiza el rendimiento de los modelos paralelos en términos de calidad de las soluciones usando métricas de cobertura de conjuntos (SC) e hiper-volumen (H). La versión multi-arranque se compara con PSA, y el modelo basado en islas se evalua con diferentes valores de mR. Además se incluye una comparativa global incluyendo todas las versiones paralelas. La evaluación de las ventajas de la paralelización en términos de speedup corresponde a la segunda parte de este análisis.

La Tabla 6.2 muestra la calidad de las soluciones no dominadas obtenidas por MS usando np={4,16} procesadores en comparación con PSA secuencial utilizando la métrica de cobertura de conjuntos SC. La primera parte de la tabla muestra los ratios de cobertura en cada grafo particionado, mientras que la segunda parte muestra el ratio medio obtenido por los seis grafos utilizados. En media, las soluciones no dominadas obtenidas por la versión secuencial de PSA dominan al 79.8 % y 82.4 % de las soluciones no dominadas obtenidas por MS_{np4} y MS_{np16}, respectivamente. Por otro lado, las soluciones no dominadas de PSA son dominadas solamente por el 8.1 % y 6.3 % de las soluciones no dominadas de MS_{np4} y MS_{np16}. Así pues, la reducción en el tiempo de ejecución obtenida por el paradigma multi-arranque conlleva una pérdida importante en la calidad de las soluciones. Esto es debido a que el rendimiento de optimización en las estrategias basadas en enfriamiento simulado depende del esquema de enfriamiento, y aunque se incremente el número de ejecuciones multi-arranque, cada una de ellas se ejecutará durante menos iteraciones, y por tanto el esquema de enfriamiento será más rápido que en la versión estándar, y también menos efectivo.

Una de las ventajas del modelo de islas radica en el bajo número de comunicaciones entre procesadores. Sin embargo, como explicamos anteriormente, la calidad de las soluciones y los tiempos de ejecución pueden variar en función del ratio de migración mR. Los resultados obtenidos (ver Tabla 6.3) indican que valores más altos de mR (migraciones cada 100 iteraciones o más) permiten una mejora en la calidad de las soluciones dominadas. En particular, cuando la migración se ejecuta cada 100 iteraciones (mR=100) las soluciones no dominadas obtenidas (I_{100}) dominan al 54.7 %, 32.4 %, y 49.5 % de las soluciones no dominadas por I_5, I_{20}, y I_{920}, mientras que los frentes obtenidos por estas últimas

Tabla 6.2: Comparativa de *MS* y *PSA* usando la métrica de cobertura de conjuntos *SC* (P_{tam}=32,np=16).

	en cada grafo						en media (AVG)		
	PSA		MS_{np4}		MS_{np16}		*PSA*	MS_{np4}	MS_{np16}
PSA	add20	3elt	1.000	1.000	1.000	1.000		0.798	0.824
	uk	add32	0.444	0.800	0.286	0.800			
	crack	wing	0.875	0.667	1.000	0.857			
MS_{np4}	0.000	0.000			0.500	0.857	0.081		0.657
	0.375	0.000			0.429	0.400			
	0.000	0.111			0.900	0.857			
MS_{np16}	0.000	0.000	0.000	0.000			0.063	0.122	
	0.375	0.000	0.333	0.400					
	0.000	0.000	0.000	0.000					

obtienen al menos una solución que domina a I_{100} en el 18.2 %, 36.0 % y 30.2 % de los casos. Aunque I_{100} e I_{920} obtienen resultados muy aproximados, I_{100} es sensiblemente superior.

Una vez hemos evaluado diferentes alternativas en el modelo multi-arranque (*MS*) y de islas (*I*) se van a comparar los mejores resultados obtenidos frente al modelo maestro-esclavo tradicional (*MW*) y el modelo de islas que utiliza división del espacio objetivo entre procesadores (I_{deo}). La Tabla 6.4 muestra la comparativa en la métrica de cobertura de conjuntos (*SC*) para las cuatro implementaciones paralelas. Las soluciones no dominadas de I_{deo} mejoran a las obtenidas por los otros modelos. Por otro lado, los conjuntos de soluciones no dominadas del modelo maestro-esclavo (*MW*) y del modelo de islas básico (I_{100}) mejoran a los obtenidos por el multi-arranque (*MS*).

Además de la métrica de cobertura de conjuntos (*SC*), hacemos uso de la métrica de hiper-volumen (*H*). La Tabla 6.5 muestra el hiper-volumen que encierra cada frente de soluciones obtenido por las cuatro alternativas de paralelización consideradas, esto es, *MS*, *MW*, I_{100}, y I_{deo}. Como se puede observar I_{deo} es el que mejores resultados obtiene para los seis grafos de prueba analizados, lo cual refuerza las conclusiones obtenidas mediante la cobertura de conjuntos (*SC*).

Otro aspecto importante a analizar es la mejora en la calidad de las soluciones cuando se incrementa el tamaño de la población. La Figura 6.12 muestra los conjuntos no dominados obtenidos por I_{deo} utilizando poblaciones de tamaño

Tabla 6.3: Impacto de mR en el modelo de islas utilizando la métrica de cobertura de conjuntos (SC) (P_{tam}=32, np=16).

	en cada grafo								en media (AVG)			
	I_5		I_{20}		I_{100}		I_{920}		I_5	I_{20}	I_{100}	I_{920}
I_5			0.500	0.000	0.333	0.000	0.000	0.000		0.158	0.182	0.151
			0.000	0.200	0.167	0.500	0.167	0.286				
			0.250	0.000	0.091	0.000	0.455	0.000				
I_{20}	0.000	0.778			0.333	0.000	0.000	0.000				
	0.667	0.333			0.000	0.667	0.167	0.429	0.502		0.360	0.336
	0.455	0.778			0.273	0.889	0.545	0.875				
I_{100}	0.000	0.889	0.000	0.875			0.000	0.857				
	0.556	0.333	0.571	0.000			0.833	0.143	0.547	0.324		0.495
	0.727	0.778	0.500	0.000			0.636	0.500				
I_{920}	0.500	0.667	0.500	0.750	0.667	0.000						
	0.556	0.500	0.571	0.400	0.000	0.667			0.561	0.433	0.302	
	0.364	0.778	0.375	0.000	0.364	0.111						

P_{tam}={32,128}. Como se puede observar, las soluciones obtenidas con P_{tam}=128 mejoran sensiblemente a las obtenidas con P_{tam}=32, lo cual es totalmente lógico ya que el incremento en el número de soluciones permite una mejor exploración del espacio de soluciones, aunque a costa de incrementar el tiempo de ejecución.

La segunda parte del análisis está enfocado a determinar qué modelo obtiene el mejor speedup. A diferencia de PrMSATS y PMSATS, donde el objetivo a minimizar es el número de cortes cumpliendo la restricción de desbalanceo, en pPSA se minimizan ambos objetivos. Esto provoca que el cálculo del speedup tenga que ser modificado con respecto a PrMSATS y PMSATS. En concreto, el criterio que vamos a considerar es el tiempo requerido por la versión paralela en encontrar la misma solución que la versión secuencial (pPSA) en términos de número de cortes. Aunque esta simplificación distorsiona claramente las conclusiones sobre speedup, es la única que permite realizar este ana.

La Figura 6.13 ofrece información en dos sentidos: la primera conclusión que alcanzamos es que la paralelización basada en islas con división del espacio objetivo (I_{deo}) obtiene los mejores resultados en speedup, mientras que MW es el peor (ver Figura 6.13(izq)). La razón de esta diferencia radica en la disparidad en el coste de comunicación entre diferentes aproximaciones. Esta separación se acentua cuando el número de procesadores se incrementa. Los resultados

Tabla 6.4: Comparativa de todas las implementaciones paralelas utilizando la métrica de cobertura de conjuntos (SC) (Ps=32, np=16).

	en cada grafo								en media (AVG)			
	MS		MW		I_{100}		I_{deo}		MS	MW	I_{100}	I_{deo}
MS			0.000	0.000	0.333	0.000	0.00	0.00		0.058	0.083	0.000
			0.250	0.000	0.167	0.000	0.00	0.00				
			0.100	0.000	0.000	0.000	0.00	0.00				
MW	1.000	0.857			0.667	0.286	0.00	0.20	0.760		0.376	0.033
	0.286	0.800			0.333	0.667	0.00	0.00				
	0.900	0.714			0.300	0.000	0.00	0.00				
I_{100}	0.000	0.857	0.000	0.167			0.00	0.00	0.593	0.161		0.000
	0.286	0.800	0.500	0.000			0.00	0.00				
	0.900	0.714	0.300	0.000			0.00	0.00				
I_{deo}	0.500	1.000	0.667	0.333	0.667	0.143			0.858	0.387	0.385	
	1.000	0.800	0.500	0.286	0.333	0.333						
	1.000	0.850	0.200	0.333	0.273	0.560						

para I_{100} son levemente peores que en I_{deo} debido a que en I_{deo} cada procesador trabaja con su sub-población en la región asignada y las comunicaciones entre procesadores se llevan a cabo solamente en la última iteración. La segunda conclusión que se puede obtener de esta figura es que usar poblaciones de gran tamaño conlleva una pérdida de speedup ya que el tiempo de espera entre comunicaciones es mayor. Como se puede observar en el análisis de eficiencia (ver Figura 6.13(der)), este inconveniente es mayor en el modelo maestro-esclavo, debido a que el número de comunicaciones es mayor que en el resto de modelos. En todo caso, y al igual que ocurría en PrMSATS y PMSATS, existe una gran irregularidad en la tendencia de ambas medidas provocada a su vez por la irregularidad del problema y su alta dependencia de factores tan simples como la elección del vértice inicial a la hora de obtener la primera partición.

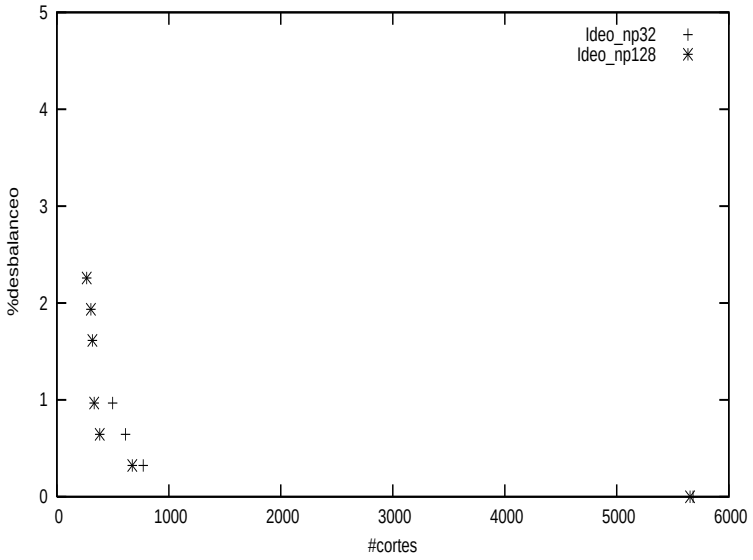

Figura 6.12: Conjuntos no dominados obtenidos con diferentes tamaños de población en *add32* (P_{tam}={32,128}, np=16).

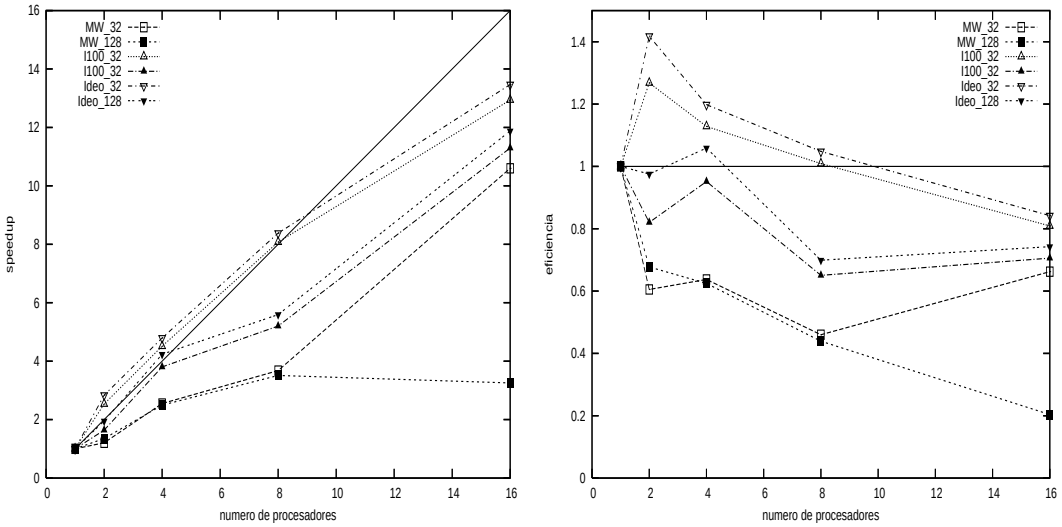

Figura 6.13: Speedup y eficiencia utilizando diferentes tamaños de población en *vibrobox* con SG=4.

Tabla 6.5: Hiper-volumen (H) obtenido por las MOMHs en todas las redes.

	MS	MW	I_{100}	I_{deo}
add20	0.0000	0.1209	0.0595	**0.4319**
3elt	0.2431	0.4386	0.4274	**0.7492**
uk	0.3368	0.2722	0.2654	**0.5636**
add32	0.1508	0.1968	0.2442	**0.8469**
crack	0.2353	0.4959	0.3178	**0.6579**
wing_nodal	0.1004	0.1041	0.2178	**0.6767**

6.6. Conclusiones

El procesamiento paralelo y distribuido se ha convertido en una herramienta muy eficaz en la ejecución de múltiples problemas de optimización, especialmente aquellos que, debido a su extrema complejidad, resultan inabordables en estaciones de trabajo, bien sea porque los recursos de memoria excedan los disponibles en una estación de trabajo, o bien porque el tiempo requerido para obtener dichas soluciones no resulta efectivo.

En este capítulo se han paralelizado tres meta-heurísticas, dos mono-objetivo y una multi-objetivo. Debido a que el problema sobre el cual se ha realizado la paralelización ha sido la partición de grafos, y debido al carácter estático de la misma, dichas paralelizaciones no se han enfocado a reducir el tiempo de ejecución de las versiones secuenciales, sino a incrementar la calidad de las soluciones haciendo uso de la diversidad en la búsqueda que se consigue con el procesamiento paralelo cooperativo. No obstante, también se observan ganancias de velocidad, que si bien no se aproximan a la linealidad, justifican el uso de su aplicación. Los resultados obtenidos denotan el buen comportamiento del modelo de islas debido a que incrementa la diversidad en la búsqueda, con las ventajas que esto conlleva en problemas tan complejos como la repartición de grafos.

En el siguiente capítulo se presentan las principales conclusiones y aportaciones de este y otros capítulos.

Conclusiones

En esta tesis doctoral se han diseñado, implementado y evaluado diferentes heurísticas híbridas para resolver problemas de optimización combinatoria de gran complejidad en el ámbito de la partición de grafos y del diseño óptimo de redes de distribución de agua, ambos problemas de gran interés y aplicabilidad. Dichos problemas son tratados mediante varios procedimientos heurísticos, incluyendo implementaciones mono-objetivo, multi-objetivo y procesamiento paralelo.

Los dos primeros capítulos se han centrado en ofrecer conceptos generales de optimización en los contextos mono-objetivo y multi-objetivo. Además, se han descrito brevemente las principales técnicas heurísticas encontradas en la literatura, algunas de las cuales han sido utilizadas con posterioridad para comparar el rendimiento de los algoritmos presentados.

En el tercer capítulo se han descrito los problemas que se han tratado en esta tesis. En concreto se trata del problema de repartición de grafos, y del problema del diseño de redes de distribución de agua malladas. Para ámbos problemas se han descrito formulaciones mono-objetivo y multi-objetivo, los cuales han sido resueltos por los métodos propuestos en los Capítulos 4, 5, y 6, así como por adaptaciones de algunos de los métodos propuestos por otros autores y descritos en los Capítulos 1 y 2. Con la idea de obtener robustas conclusiones, en ambos problemas se han utilizado diferentes instancias de prueba de diferente tamaño y utilidad. Además se han utilizado métricas de evaluación del rendimiento acordes al contexto de trabajo.

Una vez tratados los diferentes aspectos de optimización, técnicas existentes, así como los problemas de prueba a resolver, los Capítulos 4, 5, y 6 presentan los métodos propuestos en esta tesis para su resolución. En concreto, dichos algoritmos heurísticos se organizan como se indica a continuación. En el Capítulo 4 se presentan tres heurísticas híbridas de optimización mono-objetivo, dos de los cuales (rMSATS y MLrMSATS) corresponden a heurísticas híbridas para

resolver el problema de repartición de grafos, y una tercera (SSSA) que ha sido evaluada en el problema de diseño de redes de distribución de agua.

El primero de los procedimientos mono-objetivo que se ha propuesto es rMSATS, que consiste en la adaptación de un método previo, MSATS. Mientras MSATS fue diseñado para reparticionar circuitos, rMSATS generaliza su funcionamiento para repartición de grafos. Además, se ha propuesto su extensión haciendo uso del paradigma multi-nivel, creando MLrMSATS. Los resultados obtenidos, al ser comparados con algoritmos propuestos en la literatura han demostrado que rMSATS se aproximaba e incluso mejoraba alguno de los resultados encontrados por procedimientos propuestos en la bibliografía. Sin embargo, es la extensión multi-nivel, MLrMSATS, la que obtiene los mejores resultados en numerosos casos. Esta mejora respecto a rMSATS se debe a las ventajas que proporciona el esquema multi-nivel en problemas de repartición. MLrMSATS constituye un avance importante en este tipo de técnicas multi-nivel ya que la mayoría de procedimientos propuestos por otros autores utilizan técnicas de búsqueda local simples en la fase de refinamiento del paradigma multi-nivel, mientras que MLrMSATS aplica rMSATS, lo cual mejora la calidad de las soluciones al permitir escapar de óptimos locales. Además, MLrMSATS no sólo presenta como novedad el uso de este esquema de refinamiento, sino que además implementa políticas eficientes en las fases de agrupamiento y de repartición inicial que permiten mejorar la eficiencia de su rendimiento. Los resultados obtenidos por MLrMSATS conllevan a varias conclusiones claras. La primera de ellas es que en la fase de agrupamiento resulta más eficiente el uso de la estrategia de agrupamiento determinista (HEM) que la estrategia aleatoria (RM). Por otro lado, el uso de diferentes vértices de partida a la hora de obtener la repartición inicial permite mejorar la calidad de las soluciones, gracias a la diversidad que esto ofrece en el proceso de búsqueda. En la fase de refinamiento, la calidad de las soluciones es directamente proporcional al número de niveles en los cuales se aplique rMSATS.

Otro procedimiento híbrido para optimización mono-objetivo aquí propuesto es SSSA. Muchos de los algoritmos de búsqueda dispersa (SS) utilizan métodos de mejora simples. Sin embargo, SSSA aplica SA durante esta fase con el objetivo de escapar de óptimos locales. El rendimiento de SSSA ha sido evaluado en el problema diseño de redes de distribución de agua. Los resultados obtenidos, al ser comparados con otros procedimientos heurísticos, demuestran que SSSA se encuentra posicionada en una situación intermedia, siendo mejorada por un algoritmo genético, similar a enfriamiento simulado, y mejorando a otros métodos

como la búsqueda local iterada.

En cuanto a optimización multi-objetivo, el Capítulo 5 describe las dos meta-heurísticas propuestas. La primera de ellas, MOSATS, es una nueva meta-heurística poblacional que también combina enfriamiento simulado y búsqueda tabú en el proceso de optimización. MOSATS trabaja con dos poblaciones, una principal sobre la cual se lleva a cabo el proceso de optimización, y otra secundaria, o archivo externo, la cual se utiliza para almacenar las soluciones de calidad obtenidas en la población principal y que pudieran perderse durante el proceso de búsqueda. MOSATS incluye un novedoso esquema probabilístico de aceptación de soluciones basado en aceptar soluciones incomparables en función de la distribución de soluciones en el espacio objetivo. MOSATS utiliza una nueva y rápida estrategia de almacenamiento de soluciones en el archivo externo. MOSATS ha sido implementada en los dos problemas de prueba tratados, obteniendo buenos resultados en ambos. En concreto, los resultados obtenidos en el problema de repartición de grafos indican que la sinergia producida por el uso conjunto de enfriamiento simulado y búsqueda tabú permite una mejora en la calidad de las particiones con respecto a su uso separado, y en comparación con otros procedimientos existentes en la literatura. En concreto, y utilizando las métricas de cobertura de conjuntos e hiper-volumen, en media y en la mayoría de los grafos de prueba, MOSATS mejora a SMOSA, UMOSA, PSA y MOTS. Por otro lado, también hemos adaptado MOSATS al problema de diseño de redes de distribución de agua, observando que, al igual que el resto de meta-heurísticas basadas en enfriamiento simulado, se encuentra en una posición intermedia entre algoritmos evolutivos multi-objetivo como SPEA2, y otras técnicas de búsqueda local como PAES.

El segundo procedimiento propuesto, denominado MOSSSA, extiende el procedimiento mono-objetivo SSSA al contexto multi-objetivo. Además de extender la optimización a varios objetivos simultáneamente, al igual que en MOSATS, MOSSSA incluye una población secundaria de soluciones (archivo externo) para almacenar las soluciones más prometedoras encontradas en la población inicial. Los resultados obtenidos por MOSSSA en el problema del diseño de redes de distribución de agua son similares a los obtenidos por el resto de técnicas basadas en enfriamiento simulado en las redes de tamaño pequeño y mediano. En el caso de la red de gran escala (Balerma), estas técnicas se encuentran sensiblemente por detrás de los algoritmos evolutivos multi-objetivo, como SPEA2 y PESA, aunque mejorando a otras como PAES. La razón por la cual tanto MOSATS como MOSSSA son mejorados por SPEA2 o PESA radica en que estos últimos

si pueden aplicar el operador de cruce de forma eficiente en este problema, cosa que no ocurría en repartición de grafos, dada las características de ese problema.

El Capítulo 6 se presentan diferentes paralelizaciones de algoritmos propuestos en esta tesis, y también de procedimientos heurísticos propuestos por otros autores. Las dos primeras corresponden a paralelizaciones de rMSATS (PrMSATS) y MLrMSATS (PMSATS), ambas técnicas mono-objetivo. Por otro lado, la última (pPSA) corresponde a la paralelización de PSA, una meta-heurística multi-objetivo descrita en el Capítulo 2. Todas ellas han sido evaluadas en el problema de repartición de grafos. Dichas paralelizaciones están enfocadas no tanto a reducir el tiempo de ejecución, sino a incrementar la calidad de las soluciones, gracias a la diversidad que se consigue al explorar diferentes áreas del espacio de búsqueda haciendo uso del procesamiento paralelo cooperativo.

El esquema de paralelización de rMSATS y MLrMSATS es similar en ambos casos. La filosofía es la de mejorar la calidad de las soluciones mediante la exploración cooperativa de diferentes áreas del espacio de búsqueda haciendo uso del modelo de islas en diferentes procesadores. En el caso de rMSATS, cada procesador obtiene una partición inicial diferente tras aplicar el algoritmo de crecimiento de grafo (FGA) partiendo de vértices aleatorios definidos en un rango determinado por su identificador de proceso, lo cual ofrece una gran diversidad. Tras esto, se aplica el refinamiento a cada solución localizada en los diferentes procesadores, haciendo uso de aspectos cooperativos. Concretamente, se permite migrar ciertas soluciones entre procesadores con el objetivo de introducir elitismo en la búsqueda, para lo cual se han implementado tres políticas de migración diferentes. Los resultados obtenidos han demostrado la mejora que conlleva PrMSATS con respecto a rMSATS gracias a esta paralelización.

En el caso de PMSATS el planteamiento es similar. Cada procesador aplica la fase de agrupamiento partiendo de vértices distintos, al igual que la partición inicial, lo cual incrementa aun más la diversidad entre las soluciones localizadas en los diferentes procesadores. Tras esto, cada procesador aplica la fase de refinamiento con unos parámetros de enfriamiento diferentes, con el objetivo de eliminar el efecto del esquema de enfriamiento provocado por la irregularidad de los grafos de prueba. Al igual que en PrMSATS, se permiten comunicaciones esporádicas durante el proceso de búsqueda con el objetivo de introducir un cierto grado de elitismo que permita continuar la búsqueda por aquellas áreas del espacio de soluciones más prometedoras, representadas por las soluciones de mayor calidad. Los resultados obtenidos en el problema de repartición de grafos han mejorado, no sólo a rMSATS, sino también a aquellos obtenidos por PrMSATS,

lo cual demuestra la gran eficiencia de dicha paralelización multi-nivel.

Por último, se ha realizado un estudio de paralelización de PSA. A diferencia de las paralelizaciones de rMSATS y MLrMSATS (ambas heurísticas mono-objetivo), PSA es una técnica multi-objetivo poblacional. Además, mientras en los casos anteriores se aplicaba directamente el modelo de islas con el objetivo prioritario de mejorar la calidad de las soluciones, la paralelización pPSA analiza otras estrategias de paralelización alternativas. En concreto se han implementado cuatro paradigmas de paralelización: el modelo multi-arranque, el modelo maestro-esclavo, el modelo de islas, y el modelo de islas con división el espacio objetivo. De los resultados obtenidos en la formulación multi-objetivo del problema de repartición de grafos se obtienen diferentes conclusiones, como el hecho de que el modelo multi-arranque conlleva un deterioro en la calidad de las soluciones obtenidas. Esto es debido a que al lanzar más ejecuciones con menos iteraciones el proceso de enfriamiento es más rápido, lo cual conlleva un deterioro en la calidad de las soluciones. El modelo maestro-esclavo mantiene la secuencialidad del algoritmo base, PSA, aunque la calidad de las particiones es inferior que las obtenidas por los modelos de islas. El modelo de islas basado en dividir el espacio de soluciones obtiene resultados sensiblemente mejores a los obtenidos por el modelo de islas tradicional. Los resultados obtenidos también denotan que el uso de más procesadores permite mejorar la calidad de las soluciones en los modelos de islas, aunque la eficiencia tiende a reducirse cuando aumenta el número de procesadores.

La Figura 6.14 resume todas las técnicas heurísticas diseñadas e implementadas en esta tesis. Como se observa, a partir de tres meta-heuísticas como enfriamiento simulado (SA), búsqueda tabú (TS) y búsqueda dispersa (SS) se han ido construyendo dichos procedimientos. Dependiendo de la técnica en cuestión, se ha hecho uso del paradigma multi-nivel, de la optimización basada en frentes de Pareto, o del procesamiento paralelo.

Los resultados y conclusiones obtenidas en esta tesis pueden resultar de gran utilidad en diferentes ámbitos de investigación. Por un lado, en cuanto a técnicas de optimización, los nuevos procedimientos propuestos pueden servir de partida a la hora de que otros autores implementen técnicas similares haciendo uso de algunas ideas propuestas en esta tesis, sobre todo en lo referente a combinar diferentes métodos con el objetivo de mejorar el proceso de búsqueda. En cuanto a las paralelizaciones también se pueden obtener conclusiones sobre qué paradigmas de paralelización utilizar en este tipo de heurísticas.

Esta tesis también resulta de gran utilidad en lo referente a los problemas

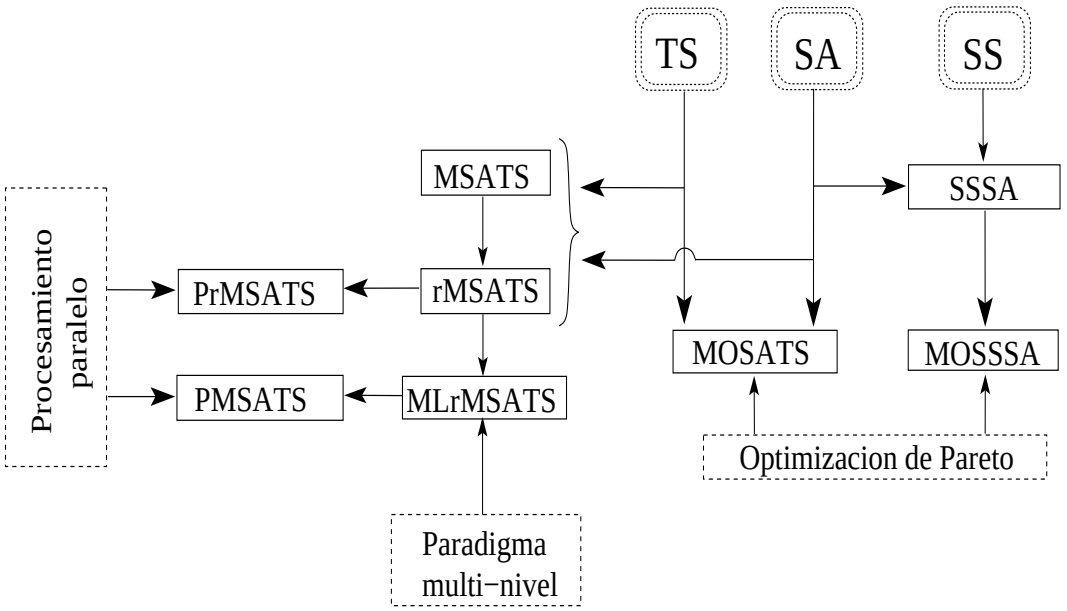

Figura 6.14: Resumen de las heurísticas diseñadas e implementadas en esta tesis.

tratados. En el caso del problema de repartición de grafos las técnicas aquí implementadas pueden ser aplicadas para resolver aplicaciones reales que se puedan modelar haciendo uso de grafos, incluyendo uno o varios objetivos. Además, las técnicas adaptadas al problema del diseño de redes de distribución de agua también resultan de gran utilidad para investigadores en la materia, ya que apenas se ha abordado el estudio de técnicas heurísticas para resolver este problema en el contexto multi-objetivo.

Conclusions

In many applications decision problems occur which can be modelled as mathematical programs using discrete decision variables that are required to satisfy a set of constraints, and an objective function that must be optimized. The aim of this dissertation thesis is the design, implementation, and evaluation of a large variety of new heuristic methods for complex computational optimization problems.

Chapters 1 and 2 have given some optimization concepts for single-objective and multi-objective optimization. Chapter 2 has also provided a global overview of the state of the art trend, including a brief description of the main methods found in the literature. Some of these methods have been adapted to the test problems to establish a thorough comparison with the new techniques proposed in this thesis.

Chapter 3 has described the test problems treated in this dissertation. Firstly graph partitioning problem, which consists of partitioning a given graph into several sub-graphs such that the number of edges connecting vertices of different sub-domains is minimized, while the imbalance is maintained below a given threshold (single-objective constrained optimization) or also minimized (multi-objective optimization). The second test problem is the looped water distribution network design, which consists of finding the best way to convey water from the sources to the users, satisfying their requirements. For both problems, single-objective and multiple-objective formulations are used in order to evaluate single-objective and multi-objective meta-heuristics. Furthermore, instances of different sizes and topologies are used in both problems.

Chapters 4, 5, and 6 have described all the meta-heuristic algorithms proposed in this thesis. These have been organized in the following way: Chapter 4 has presented three single-objective optimization algorithms, two of them, rMSATS and MLrMSATS are hybrid heuristics to solve graph partitioning problems, and the third one, SSSA, is a meta-heuristic, which has been evaluated in the wa-

ter distribution problem; Chapter 5 has described two multi-objective hybrid meta-heuristics, MOSATS and MOSSSA, which have been applied to both test problems; Chapter 6 has introduced the parallelizations of three algorithms: rMSATS, MLrMSATS, and PSA. Parallel implementations have been focused on increasing the quality of the solutions by means of cooperation processes.

The first single-objective procedure presented in Chapter 4 is rMSATS. It consists of the adaptation of a previous method, MSATS, to solve the graph partitioning problem. The results obtained have demonstrated the good performance of rMSATS in comparison with other methods found in the literature. rMSATS has been extended making use of the multi-level paradigm, creating MLrMSATS. The results obtained by MLrMSATS outperform rMSATS in most cases. This improvement is due to the advantages of using the multi-level scheme in partitioning problems. MLrMSATS constitutes an important advance in multi-level techniques for graph partitioning because, thus far, almost all the methods found in the literature use simple local search techniques, such as the Kernighan and Lin heuristic, in the refinement phase, while MLrMSATS applies rMSATS, which allows the quality of the solutions to be increased. Moreover, MLrMSATS increases its efficiency thanks to the use of efficient strategies in the coarsening and first partitioning phases. For instance, the deterministic matching strategy (HEM) outperforms random matching (RM). On the other hand, using random vertices in the coarsening and initial partitioning phases also allows an improvement in the solutions. Another conclusion obtained is that the quality of the solutions obtained by MLrMSATS is proportional to the number of uncoarsening (refinement) levels where rMSATS is applied.

The third single-objective meta-heuristic proposed in Chapter 4 is SSSA, which combines scatter search (SS) with simulated annealing (SA). Most computational optimization techniques that make use of SS use simple methods in the optimization phase. However, SSSA extends it, using SA to escape from local optima. The performance of SSSA has been evaluated in the water distribution problem, and the results obtained indicate that it is ranked in an intermediate position in comparison with other methods found in the literature. In particular, SSSA is outperformed by a genetic algorithm, obtain similar results than simulated annealing, and improves to iterated local search.

Chapter 5 presents two new meta-heuristics for multi-objective optimization. The first one, MOSATS, is a population-based algorithm that also uses a secondary population, called external archive, to store the promising solutions found in the main population. In addition to hybridizing simulated annealing and tabu

search, MOSATS introduces a novel selection scheme based on accepting incomparable solutions according to the distribution of the other ones in the search space. The external archive is managed using a new and fast storing strategy. MOSATS has been evaluated in both test problems obtaining good results. The results obtained in the graph partitioning problem indicate that the synergy obtained by combining SA and TS allows an improvement in the quality of the partitions in comparison with their separate use. MOSATS has also been compared with other methods found in the literature, using the set coverage and hyper-volume metrics. The results obtained have demonstrated that MOSATS outperforms SMOSA, UMOSA, PSA and MOTS. On the other hand, MOSATS has also been adapted to the water distribution network design problem, and like other annealing-based methods it ranked in an intermediate position, improving techniques such as PAES, but being outperformed by multi-objective evolutionary algorithms such as SPEA2.

The second multi-objective procedure proposed is MOSSSA, which extends the single-objective procedure SSSA to the multi-objective context. Like other multi-objective approaches, MOSSSA also uses two populations: a main population and an external archive of non-dominated solutions. The results obtained by MOSSSA in the water distribution network problem indicate that, like MOSATS, it is ranked in an intermediate position. In particular, in small and intermediate networks all the methods are very close, while in large networks (Balerma) the best performance is obtained by pure multi-objective evolutionary approaches (SPEA2 and PESA). The reason behind these results is that, contrary to the graph partitioning problem, evolutionary operators like crossover can be applied without problems to the water distribution problem.

After describing the performance of single-objective and multi-objective metaheuristics, Chapter 6 have presented three parallel implementations. The first and second parallelizations correspond to parallel versions of rMSATS (PrMSATS) and MLrMSATS (PMSATS), both single-objective algorithms. On the other hand, the last one (pPSA) parallelizes the multi-objective algorithm called Pareto Simulated Annealing (PSA). All the parallelizations have been tested in the graph partitioning problem.

The parallelization scheme in PrMSATS and PMSATS is similar in both cases. The idea is to outperform the quality of the solutions by means of cooperative exploration of different areas of the search space using several processors under the island parallelization model. In PrMSATS, each processor obtains an initial partition after applying a graph growing algorithm (FGA) starting from

a random vertex, which provides diversity to the search process. After that, each processor applies rMSATS using its own annealing scheme. In some iterations, elitism is introduced by means of migrating the best solutions found in each processor. This cooperative process is performed by means of several migration policies. The results obtained have demonstrated the improvement provided by PrMSATS in reference to rMSATS thanks to this parallelization strategy.

On the other hand, PMSATS works in a similar way to PrMSATS. In this case, each processor applies the coarsening phase starting from random vertices, the same as the initial partition, which increases the diversity among the solutions found in different processors even more. Each processor then applies the uncoarsening/refinement process using its own annealing values. Like PrMSATS, PMSATS also allows cooperation among processors by means of migrating the best solutions found in the search using several migration policies. The results obtained in the graph partitioning problem have improved, not only with rMSATS, but also with PrMSATS, which reinforces the conclusions about the great efficiency of this parallelization.

Finally, the third parallelization is related to a multi-objective algorithm, like PSA. While PrMSATS and PMSATS applied the island parallelization paradigm directly, pPSA analyzes several parallelization techniques found in the literature, including the multi-start model, the master-slave paradigm, the standard island model, and a recent extension of the island model for multi-objective optimization that divides the objective space in areas in such a way that each processor works in one specific region. The results obtained in the multi-objective formulation of the graph partitioning problem offers some conclusions. The first one is that the multi-start model does not provide good results as it performs more runs with less iterations per run. This means that the annealing schedule is faster, and therefore the quality of the solutions poorer. The master-slave paradigm maintains the sequentiality of PSA, but the quality of the solutions is worse than using the island model. In particular, the island model with division of the search space among processors obtains solutions slightly better than the standard island paradigm. Finally, it has been verified that the quality of the partitions improves when the number of processors increases. However, the efficiency decreases because there are more breaks in communication.

Figure 6.14 summarizes the heuristic methods proposed in this thesis. As it can be seen, three meta-heuristics are used: simulated annealing (SA), tabu search (TS), and scatter search (SS). Furthermore, according to the particular implementation, some other tools are used, such as the multi-level paradigm,

Pareto-based optimization, or parallel processing.

The results and conclusions obtained in this dissertation can be very useful in different fields. In reference to the computational optimization techniques, the new procedures here presented can be used by other authors to define their own algorithms, taking ideas from them, for instance, in reference to the advantages of combining several optimization techniques. On the other hand, the multi-objective approaches here presented constitute interesting alternatives to other local search-based multi-objective methods. In reference to parallel algorithms, the results indicate that in local-search based methods the island paradigm works better than other approaches. The results obtained are also very useful in terms of the problems treated. Researchers who work with real problems that can be modelled by graphs, or engineers working with water distribution systems can directly adapt the algorithms here proposed to their particular problems.

Apéndice A

Tablas de resultados (GPP)

Tabla A.1: Grafos de prueba.

grafo	|V|	|E|	min	max	avg	tam. arch. (KB)
c432	292	432	2	4	2.96	3
c1355	878	1355	1	13	3.09	10
c6288	3936	6288	1	17	3.20	58
add20	2395	7462	1	123	6.23	63
data	2851	15093	3	17	10.59	140
3elt	4720	13722	3	9	5.81	136
uk	4824	6837	1	3	2.83	70
add32	4960	9462	1	31	3.82	90
whitaker3	9800	28939	3	8	5.92	294
crack	10240	30380	3	9	5.93	297
wing_nodal	10937	75488	5	28	13.80	768
fe_4elt	11143	32818	3	12	5.89	341
vibrobox	12328	165250	8	120	26.81	1679
bcsstk29	13992	302748	4	70	43.27	1679
4elt	15606	45878	3	10	5.88	501
fe_sphere	16386	49152	4	6	6.00	540
cti	16840	48232	3	6	5.73	532
memplus	17758	54196	1	573	6.10	536
cs4	22499	43858	2	4	3.90	506
bcsstk30	28924	1007284	3	218	69.65	11403
bcsstk31	35588	572914	1	188	32.20	6547
bcsstk32	44609	985046	1	215	44.16	11368
t60k	60005	89440	2	3	2.98	1100
wing	62032	121544	2	4	3.92	1482
brack2	62631	366559	3	32	11.71	4358
finan512	74752	261120	2	54	6.99	3128
fe_tooth	78136	452591	3	39	11.58	5413
fe_rotor	99617	662431	5	125	13.30	7894
598a	110917	741934	5	26	13.37	9030
fe_ocean	143437	409593	1	6	5.71	5242
wave	156317	1059331	3	44	13.55	13479
m14b	214765	1679016	4	40	15.64	21996

Tabla A.2: Resultados obtenidos por rMSATS frente a METIS (1/4).

grafo	algoritmo	*SG* = 2		*SG* = 4		*SG* = 8		*SG* = 16		*SG* = 32	
		cortes	desb.	cortes	desb.	cortes	desb.	cortes	desb.	cortes	desb.
c432	pMETIS	36	0	61	0	80	1	119	10	171	21
	kMETIS	39	1	60	3	87	1	147	37	179	64
	FGA	98	0	291	0	363	0	397	0	415	0
	rMSATS	**33**	1	**56**	1	**76**	3	**103**	0	**147**	0
c499	pMETIS	23	0	61	1	92	5	140	5	188	5
	kMETIS	**20**	4	62	2	92	3	167	20	211	211
	FGA	109	0	268	0	387	0	440	0	473	0
	rMSATS	**20**	4	**55**	5	**83**	5	**131**	5	**185**	1
c880	pMETIS	24	0	60	1	100	1	153	5	211	8
	kMETIS	44	3	60	3	93	2	152	2	219	24
	FGA	135	0	331	0	596	0	746	0	802	0
	rMSATS	**19**	1	**50**	5	**84**	4	**132**	3	**198**	0
c1355	pMETIS	24	0	**59**	0	101	0	134	4	204	6
	kMETIS	**22**	1	68	2	97	4	144	9	218	13
	FGA	305	0	627	0	974	0	1173	0	1262	0
	rMSATS	28	2	**59**	4	**81**	5	**115**	4	**184**	4
c3540	pMETIS	75	0	147	0	289	0	384	1	566	2
	kMETIS	92	3	184	3	264	3	417	3	577	2
	FGA	300	0	952	0	1890	0	2783	0	3079	0
	rMSATS	**66**	0	**141**	2	**218**	4	**329**	5	**490**	4
c5315	pMETIS	87	0	164	0	250	0	388	1	554	1
	kMETIS	**77**	1	**159**	3	273	3	387	4	554	3
	FGA	531	0	1316	0	2896	0	3980	0	4426	0
	rMSATS	83	0	169	3	**229**	4	**326**	5	**483**	5
c6288	pMETIS	153	0	257	0	384	0	504	1	632	2
	kMETIS	161	3	265	1	408	1	507	3	661	2
	FGA	1297	0	3065	0	4214	0	4913	0	5194	0
	rMSATS	**134**	0	**246**	5	**326**	2	**416**	5	**561**	5

Tabla A.3: Resultados obtenidos por rMSATS frente a METIS (2/4).

grafo	algoritmo	*SG* = 2		*SG* = 4		*SG* = 8		*SG* = 16		*SG* = 32	
		cortes	desb.	cortes	desb.	cortes	desb.	cortes	desb.	cortes	desb.
add20	pMETIS	725	0	1292	0	1907	0	2504	1	3008	3
	kMETIS	719	3	**1257**	3	1857	5	2442	7	3073	74
	FGA	1549	0	2811	0	3696	0	4577	0	4940	0
	rMSATS	**701**	3	1493	5	**1769**	5	**2157**	5	**2466**	4
uk	pMETIS	23	0	67	0	101	0	189	0	316	1
	kMETIS	36	3	**64**	3	98	2	189	3	302	3
	FGA	97	0	221	0	469	0	970	0	1971	0
	rMSATS	**30**	0	69	1	**89**	4	**161**	5	**259**	5
add32	pMETIS	**21**	0	**42**	0	**81**	0	**128**	0	**288**	1
	kMETIS	28	2	44	3	102	3	206	3	352	15
	FGA	1067	0	2968	0	5166	0	5866	0	6169	0
	rMSATS	41	1	119	5	191	5	285	5	349	5
3elt	pMETIS	108	0	231	0	388	0	665	0	1093	0
	kMETIS	97	1	213	2	403	2	651	2	1096	2
	FGA	239	0	724	0	1487	0	3015	0	6147	0
	rMSATS	**90**	0	**201**	4	**340**	5	**567**	5	**959**	5
data	pMETIS	**218**	0	480	0	842	0	1370	0	2060	1
	kMETIS	233	2	454	2	806	3	1350	3	2080	2
	FGA	300	0	770	0	1493	0	2988	0	6018	0
	rMSATS	227	2	**427**	5	**727**	5	**1168**	5	**1818**	5
whitaker3	pMETIS	135	0	406	0	719	0	1237	0	1891	0
	kMETIS	133	3	446	3	769	3	1200	3	1824	3
	FGA	304	0	728	0	1642	0	3449	0	7019	0
	rMSATS	**128**	0	**385**	5	**687**	5	**1093**	5	**1683**	5
crack	pMETIS	187	0	382	0	773	0	1255	0	1890	0
	kMETIS	225	1	408	3	809	3	1218	3	1882	3
	FGA	399	0	935	0	2048	0	4196	0	8403	0
	rMSATS	**184**	0	**363**	2	**685**	4	**1098**	5	**1689**	5

Tabla A.4: Resultados obtenidos por rMSATS frente a METIS (3/4).

grafo	algoritmo	*SG* = 2		*SG* = 4		*SG* = 8		*SG* = 16		*SG* = 32	
		cortes	desb.	cortes	desb.	cortes	desb.	cortes	desb.	cortes	desb.
4elt	pMETIS	**154**	0	**406**	0	**635**	0	1056	0	1769	0
	kMETIS	225	0	344	2	614	3	1099	3	1784	3
	FGA	735	0	1848	0	4205	0	8684	0	17445	0
	rMSATS	187	0	437	3	649	5	**945**	5	**1534**	5
cs4	pMETIS	414	0	1154	0	1746	0	2538	0	3579	0
	kMETIS	410	0	1173	3	1677	3	2521	3	3396	3
	FGA	1277	0	3489	0	7672	0	15685	0	27983	0
	rMSATS	**377**	0	**1018**	1	**1484**	4	**2245**	5	**3056**	5
cti	pMETIS	**334**	0	**1113**	0	2110	0	3181	0	4605	0
	kMETIS	395	2	1132	1	2130	2	3451	3	4713	3
	FGA	1698	0	4446	0	9548	0	19402	0	34918	0
	rMSATS	588	1	1177	5	**1853**	5	**2875**	5	**4059**	5
memplus	pMETIS	**6337**	0	10559	0	13110	0	14942	0	17303	0
	kMETIS	6453	3	**10483**	3	**12615**	3	**14604**	6	16821	6
	FGA	13433	0	23063	0	27957	0	30818	0	33417	0
	rMSATS	8570	4	12438	5	13640	5	15301	5	**16269**	5
fe_sphere	pMETIS	440	0	872	0	1330	0	2030	0	2913	0
	kMETIS	444	0	903	3	1306	3	2012	2	2842	2
	FGA	472	0	1342	0	2914	0	6050	0	12294	0
	rMSATS	**386**	0	**774**	4	**1226**	4	**1726**	5	**2511**	5
wing_nodal	pMETIS	1820	0	4000	0	6070	0	9290	0	13237	0
	kMETIS	1855	0	4355	2	6337	3	9465	3	12678	3
	FGA	5368	0	14118	0	29960	0	39425	0	44408	0
	rMSATS	**1723**	0	**3816**	5	**5419**	5	**8392**	5	**11741**	5

Tabla A.5: Resultados obtenidos por rMSATS frente a METIS (4/4).

grafo	algoritmo	*SG* = 2		*SG* = 4		*SG* = 8		*SG* = 16		*SG* = 32	
		cortes	desb.	cortes	desb.	cortes	desb.	cortes	desb	cortes	desb
wing	pMETIS	950	0	2086	0	3205	0	4666	0	6700	0
	kMETIS	**909**	1	**1943**	3	3120	3	4652	3	6613	3
	FGA	2785	0	7062	0	15375	0	31486	0	63456	0
	rMSATS	1353	0	2069	5	**2877**	5	**4254**	5	**5938**	5
vibrobox	pMETIS	12427	0	**21471**	0	**28177**	0	37441	0	46112	0
	kMETIS	11952	1	23141	2	29640	3	38673	3	45613	3
	FGA	27835	0	64754	0	88603	0	105714	0	118650	0
	rMSATS	**11604**	0	20500	5	30572	5	**36487**	5	**44678**	5
fe_ocean	pMETIS	505	0	**2039**	0	**4516**	0	9613	0	14613	0
	kMETIS	536	0	2194	2	5627	2	10253	3	16604	3
	FGA	2376	0	7938	0	17855	0	37430	0	76065	0
	rMSATS	**464**	0	2349	1	5481	5	**9051**	5	**13923**	5
fe_tooth	pMETIS	4292	0	8577	0	13653	0	**19346**	0	29215	0
	kMETIS	4262	2	**7835**	3	13544	3	20455	3	28572	3
	FGA	7785	0	32262	0	71076	0	145539	0	218804	0
	rMSATS	**4086**	0	11817	5	**13152**	5	21562	5	**26981**	5
598a	pMETIS	**2504**	0	8533	0	17276	0	28922	0	44760	0
	kMETIS	2533	0	**8495**	1	**17137**	3	28647	3	44398	3
	FGA	26129	0	69388	0	145234	0	181053	0	243981	0
	rMSATS	3881	0	12603	5	22734	5	**27976**	5	**40896**	5
wave	pMETIS	**9493**	0	23032	0	34795	0	**48106**	0	72404	0
	kMETIS	9655	0	21682	2	**33146**	3	48183	3	67860	3
	FGA	32380	0	85289	0	184382	0	381418	0	514427	0
	rMSATS	12744	0	**21390**	5	38522	5	51920	5	**64983**	5

Tabla A.6: Comparación entre MLrMSATS y METIS.

grafo	algoritmo	*SG* = 2		*SG* = 4		*SG* = 8		*SG* = 16		*SG* = 32	
		cortes	desb	cortes	desb	cortes	desb	cortes	desb	cortes	desb
add20	pMetis	725	0	1292	0	1907	0	2504	1	3008	3
	kMetis	719	3	1257	3	1857	5	2442	5	3073	74
	MLrMSATS	**696**	5	**1193**	5	**1757**	5	**2186**	5	**2793**	4
add32	pMetis	21	0	42	0	**81**	0	**128**	0	**288**	1
	kMetis	28	2	44	3	102	3	206	3	352	15
	MLrMSATS	**10**	0	**35**	3	133	5	184	4	294	5
wing nodal	pMetis	1820	0	4000	0	6070	0	9290	0	13237	0
	kMetis	1855	0	4355	2	6337	3	9465	3	12678	3
	MLrMSATS	**1670**	5	**3596**	5	**5387**	5	**8425**	5	**11812**	6
fe_4elt2	pMetis	**130**	0	359	0	654	0	1152	0	1787	0
	kMetis	132	0	398	3	684	3	1149	3	1770	3
	MLrMSATS	**130**	0	**350**	2	**632**	5	**1054**	5	**1662**	6
vibro box	pMetis	12427	0	21471	0	28177	0	37441	0	46112	0
	kMetis	**11952**	1	23141	2	29640	3	38673	3	45613	3
	MLrMSATS	12096	3	**20391**	5	**26147**	5	**34791**	5	**42530**	5
mem plus	pMetis	6337	0	10559	0	13110	0	14942	0	17303	0
	kMetis	6453	3	10483	3	12615	3	14604	6	16821	6
	MLrMSATS	**5662**	5	**9427**	5	**12283**	5	**14176**	5	**15661**	5
cs4	pMetis	414	0	1154	0	1746	0	2538	0	3579	0
	kMetis	**410**	0	1173	3	**1677**	3	2521	3	3396	3
	MLrMSATS	418	4	**1103**	4	1696	5	**2430**	5	**3371**	5
bcsstk 32	pMetis	5672	0	12225	0	**23601**	0	43371	0	**70020**	0
	kMetis	5374	0	**11561**	3	27311	3	**42581**	3	82864	3
	MLrMSATS	**5193**	5	18821	5	34596	5	51341	5	78260	7
brack2	pMetis	738	0	3250	0	**7844**	0	**12655**	0	19786	0
	kMetis	769	0	3458	1	8776	3	13652	3	19888	3
	MLrMSATS	**668**	4	**2808**	5	8582	5	13770	5	**18245**	5
fe_tooth	pMetis	4292	0	8577	0	13653	0	19346	0	29215	0
	kMetis	**4262**	2	7835	3	**13544**	3	20455	3	28572	3
	MLrMSATS	4436	5	**7152**	4	14081	5	**19201**	5	**26016**	5
fe_rotor	pMetis	2190	0	8564	0	15712	0	23863	0	36225	0
	kMetis	2294	0	8829	3	14929	3	24477	3	36159	3
	MLrMSATS	**1974**	3	**8143**	5	**14495**	4	**23807**	5	**33900**	5
fe_ocean	pMetis	505	0	**2039**	0	**4516**	0	9613	0	14613	0
	kMetis	536	0	2194	2	5627	2	10253	3	16604	3
	MLrMSATS	**317**	3	3035	5	6023	5	**8699**	5	**13954**	5
wave	pMetis	9493	0	23032	0	34795	0	48106	0	72404	0
	kMetis	9655	0	21682	2	33146	3	48183	3	67860	3
	MLrMSATS	**8868**	5	**18246**	5	**30583**	5	**44625**	5	**63809**	5

Tabla A.7: Cobertura de conjuntos (SC) de MOSATS, SMOSA, UMOSA, PSA, y MOTS en los grafos de prueba.

	MOSATS		SMOSA		UMOSA		PSA		MOTS	
	data	3elt	1.000	1.000	0.500	0.667	1.000	0.875	0.625	1.000
	uk	wing.	0.200	1.000	1.000	0.000	0.200	1.000	1.000	1.000
MOSATS	vibr.	cti	1.000	1.000	0.857	0.250	1.000	0.800	1.000	1.000
	memp.	cs4	0.722	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	0.000	0.000			0.000	0.000	0.333	0.000	0.500	0.714
	0.714	0.000			0.333	0.000	0.400	0.800	0.714	0.714
SMOSA	0.000	0.000			0.000	0.000	0.300	0.000	0.650	0.000
	0.214	0.000			0.875	0.000	0.684	0.000	0.932	0.800
	0.100	0.200	0.000	1.000			0.083	1.000	0.625	0.857
	0.000	1.000	0.200	1.000			0.200	1.000	1.000	1.000
UMOSA	0.231	0.800	0.917	1.000			0.950	1.000	0.950	1.000
	1.000	1.000	0.556	1.000			0.421	1.000	1.000	1.000
	0.000	0.200	0.255	0.857	0.000	0.000			0.500	0.857
	0.857	0.000	0.400	0.000	0.667	0.000			0.857	0.714
PSA	0.000	0.000	0.500	0.714	0.000	0.000			0.850	0.000
	0.000	0.000	0.000	0.833	0.000	0.000			0.682	0.600
	1.000	0.000	1.000	0.143	1.000	0.333	0.000	0.143		
	0.000	0.000	0.200	0.000	0.000	0.000	0.200	0.000		
MOTS	0.000	0.000	0.167	0.286	0.048	0.000	0.100	0.100		
	0.000	0.000	0.000	0.167	0.000	0.000	0.000	0.091		

Tabla A.8: Efecto de aplicar PrMSATS sobre diferentes tamaños de población.

grafo	algoritmo	*SG* = 2		*SG* = 4		*SG* = 8		*SG* = 16		*SG* = 32	
		cortes	desb.	cortes	desb.	cortes	desb	cortes	desb.	cortes	desb.
add20	PrMSATS^{1}	843	5	1349	5	1791	5	2159	5	2637	4
	PrMSATS^{5}	715	5	**1223**	5	1769	5	2128	5	2499	4
	PrMSATS^{20}	**701**	5	1232	5	**1749**	5	**2122**	5	**2481**	4
add32	PrMSATS^{1}	51	2	84	5	154	5	165	5	279	5
	PrMSATS^{5}	11	5	82	3	123	5	171	3	285	5
	PrMSATS^{20}	**10**	0	**55**	2	**88**	5	**158**	5	**243**	5
wing nodal	PrMSATS^{1}	1674	5	3626	3	5485	5	8370	5	11905	5
	PrMSATS^{5}	**1673**	5	3631	5	5442	5	8394	5	11882	5
	PrMSATS^{20}	**1673**	5	**3545**	5	**5385**	5	**8341**	5	**11831**	5
fe 4elt2	PrMSATS^{1}	206	0	385	3	752	5	1223	5	1805	5
	PrMSATS^{5}	**130**	0	**372**	4	715	5	1209	5	1767	5
	PrMSATS^{20}	**130**	0	394	3	**704**	5	**1170**	5	**1704**	5
vibro box	PrMSATS^{1}	12229	5	**20777**	5	31092	5	34460	5	42118	5
	PrMSATS^{5}	11767	5	20923	5	28688	5	34115	5	41418	5
	PrMSATS^{20}	**10714**	3	20868	5	**26201**	5	**33476**	5	**41270**	5
mem plus	PrMSATS^{1}	7600	5	10795	5	12644	5	14158	5	15492	5
	PrMSATS^{5}	7402	4	10554	5	12687	5	14014	5	**15129**	5
	PrMSATS^{20}	**7119**	5	**10401**	5	**12460**	5	**13858**	5	15306	5
cs4	PrMSATS^{1}	861	4	1554	4	2261	5	2807	5	3355	5
	PrMSATS^{5}	446	4	1094	5	1914	5	2651	5	3405	5
	PrMSATS^{20}	**421**	4	**1067**	5	**1802**	5	**2472**	5	**3287**	5
bcsstk 32	PrMSATS^{1}	10422	1	29685	5	52697	5	68424	5	86580	5
	PrMSATS^{5}	**10248**	2	**21181**	5	45154	5	70488	5	85455	5
	PrMSATS^{20}	10807	4	21406	5	**42607**	5	**61386**	5	**82726**	5
brack2	PrMSATS^{1}	711	1	3874	5	**8028**	5	14816	5	20676	5
	PrMSATS^{5}	711	2	3743	5	8490	5	14295	5	20497	5
	PrMSATS^{20}	**683**	4	**3458**	5	8304	5	**13614**	5	**20208**	5
fe tooth	PrMSATS^{1}	3966	5	10486	5	15920	5	23375	5	32624	5
	PrMSATS^{5}	**3896**	5	10332	5	16707	5	22602	5	28514	5
	PrMSATS^{20}	4054	5	**10117**	5	**15695**	4	**21596**	5	**27642**	5
fe rotor	PrMSATS^{1}	2058	2	9445	5	17735	5	28694	5	39102	5
	PrMSATS^{5}	1968	3	7632	3	**15875**	5	26222	5	38192	5
	PrMSATS^{20}	**1956**	4	**7535**	4	16896	5	**25672**	5	**37770**	5
fe ocean	PrMSATS^{1}	406	2	**2670**	5	**6583**	5	12232	5	18482	5
	PrMSATS^{5}	356	3	2856	5	7238	5	12029	5	17696	5
	PrMSATS^{20}	**317**	3	2954	5	6962	5	**11501**	5	**16030**	5
wave	PrMSATS^{1}	15496	5	30899	5	47786	5	58013	5	74920	5
	PrMSATS^{5}	10132	5	**28235**	5	44929	5	57533	5	76448	5
	PrMSATS^{20}	**8670**	5	28474	5	**39365**	5	**53437**	5	**72380**	5

Tabla A.9: Resultados obtenidos por las políticas de migración propuestas.

grafo	algoritmo	*SG* = 2		*SG* = 4		*SG* = 8		*SG* = 16		*SG* = 32	
		cortes	desb.	cortes	desb.	cortes	desb.	cortes	desb.	cortes	desb.
add20	PrMSATS^A	**684**	5	1249	5	**1772**	5	**2125**	5	2462	4
	PrMSATS^B	**684**	5	**1245**	5	1773	5	**2125**	5	**2458**	4
	PrMSATS^C	700	5	**1245**	5	1784	5	2132	5	2498	4
add32	PrMSATS^A	**11**	4	**54**	3	88	5	139	5	**238**	3
	PrMSATS^B	**11**	4	**54**	3	88	5	140	5	**238**	5
	PrMSATS^C	**11**	1	57	4	**85**	5	**136**	5	**238**	3
wing nodal	PrMSATS^A	**1670**	5	**3616**	4	5391	5	**8323**	5	11826	5
	PrMSATS^B	**1670**	5	**3616**	5	**5389**	5	8326	5	11821	5
	PrMSATS^C	1672	5	3937	5	5467	5	8372	5	**11753**	5
fe 4elt2	PrMSATS^A	**130**	0	350	2	**707**	5	1132	5	**1689**	5
	PrMSATS^B	**130**	0	350	1	**707**	5	**1111**	5	1725	5
	PrMSATS^C	**130**	0	**349**	0	781	3	1169	5	1715	5
vibro box	PrMSATS^A	**10310**	0	20545	5	27091	5	33243	5	41518	5
	PrMSATS^B	**10310**	3	20532	5	27088	5	33242	5	**41513**	5
	PrMSATS^C	10916	0	**20531**	5	**26937**	5	**33240**	5	41603	5
mem plus	PrMSATS^A	7254	5	10488	5	12291	5	13702	5	15307	5
	PrMSATS^B	7252	5	10482	5	**12286**	5	**13701**	5	**15304**	5
	PrMSATS^C	**7217**	5	**10475**	5	12360	5	14043	5	15402	5
cs4	PrMSATS^A	365	5	1042	4	1737	5	2457	5	3289	5
	PrMSATS^B	**364**	5	**1036**	4	**1727**	5	**2456**	5	3328	5
	PrMSATS^C	377	5	1041	5	1737	5	2571	5	**3318**	5
bcsstk 32	PrMSATS^A	**10545**	5	22268	5	40076	5	63267	5	82048	5
	PrMSATS^B	**10545**	5	22269	5	40076	5	63267	5	82048	5
	PrMSATS^C	**10545**	5	**21778**	5	**39902**	5	**63110**	5	**80861**	5
brack2	PrMSATS^A	672	4	**2793**	5	7808	5	13856	5	18891	5
	PrMSATS^B	672	4	**2793**	5	7803	5	**13854**	5	**18889**	5
	PrMSATS^C	**665**	4	3720	3	**7291**	5	14409	5	18954	5
fe tooth	PrMSATS^A	3878	5	9036	5	15520	5	21675	5	28414	5
	PrMSATS^B	3898	1	**8972**	5	15575	5	21670	5	**28401**	5
	PrMSATS^C	**3853**	3	10351	5	**15053**	5	**21175**	5	28842	5
fe rotor	PrMSATS^A	**1966**	3	7477	3	**15873**	5	**25585**	5	37579	5
	PrMSATS^B	1968	2	**7475**	3	15874	5	25586	5	**37547**	5
	PrMSATS^C	1967	2	8006	5	17780	5	26146	5	38874	5
fe ocean	PrMSATS^A	**317**	3	2864	5	6718	5	10697	5	17248	5
	PrMSATS^B	**317**	3	2863	5	6731	5	**10679**	5	17329	**5**
	PrMSATS^C	325	3	**2515**	5	**6272**	5	10834	5	**16593**	5
wave	PrMSATS^A	8614	5	27060	5	**38660**	5	**53520**	5	**67053**	5
	PrMSATS^B	**8613**	5	27056	5	38662	5	**53520**	5	67119	5
	PrMSATS^C	9769	1	**25355**	2	43018	5	55558	5	70019	5

Tabla A.10: Comparativa: PMSATS, ParMETIS y GPA (1/2).

grafo	algoritmo	SG=2	SG=4	SG=8	SG=16	SG=32	SG=64
		cortes	cortes	cortes	cortes	cortes	cortes
add20	PMSATS	638	**1184**	1709	**2107**	**2583**	**3182**
	ParMETIS	778	1327	2053	3060	3790	3927
	GPA	**551**	**1184**	**1705**	2186	2758	3266
data	PMSATS	189	391	**681**	**1147**	**1815**	**2803**
	ParMETIS	225	468	871	1441	4415	7668
	GPA	**181**	**378**	702	1195	1922	2911
3elt	PMSATS	**87**	**199**	336	567	**956**	**1535**
	ParMETIS	108	266	448	884	3328	4178
	GPA	**87**	**199**	**334**	**566**	958	1552
uk	PMSATS	21	47	93	166	288	466
	ParMETIS	24	75	147	334	527	795
	GPA	**18**	**41**	**82**	**154**	**265**	**436**
add32	PMSATS	**10**	36	**67**	150	246	**597**
	ParMETIS	**10**	**33**	309	462	677	1130
	GPA	**10**	**33**	69	**117**	**212**	624
whitaker3	PMSATS	126	381	**658**	1100	1698	2544
	ParMETIS	132	489	862	1404	5425	6110
	GPA	**126**	**380**	**658**	**1092**	**1686**	**2535**
crack	PMSATS	**182**	361	**676**	1083	1690	**2540**
	ParMETIS	209	412	845	1296	1988	2883
	GPA	**182**	**360**	**676**	**1082**	**1679**	2590
wing nodal	PMSATS	1669	**3564**	**5378**	8332	**11814**	**15789**
	ParMETIS	1908	3887	5929	9164	12787	17374
	GPA	**1668**	3566	5387	**8316**	12024	16102
fe_4elt2	PMSATS	**130**	**349**	601	1012	**1641**	2520
	ParMETIS	**130**	392	710	1143	1831	2796
	GPA	**130**	**349**	**597**	**1007**	1651	**2516**
vibrobox	PMSATS	10630	20050	24338	33460	41356	**47149**
	ParMETIS	12802	21239	28701	37882	45311	51552
	GPA	**10310**	**19245**	**24158**	**31695**	**41176**	50757
bcsstk29	PMSATS	**2818**	8388	**15047**	**23235**	**34843**	**56120**
	ParMETIS	2958	9617	18840	27456	41680	60938
	GPA	**2818**	**8088**	15314	24706	36731	58108
4elt	PMSATS	**137**	322	532	939	1554	2583
	ParMETIS	163	387	652	1103	1835	2938
	GPA	**137**	**319**	**527**	**916**	**1537**	**2581**
fe_sphere	PMSATS	**384**	776	1193	1719	2575	3623
	ParMETIS	458	906	1395	2081	2966	4142
	GPA	**384**	**766**	**1152**	**1692**	**2477**	**3547**
cti	PMSATS	**318**	**889**	1727	2781	**4034**	**5738**
	ParMETIS	459	1104	2212	3452	4963	6753
	GPA	**318**	917	**1716**	**2778**	4236	5907

Tabla A.11: Comparativa: PMSATS, ParMETIS y GPA (2/2).

grafo	algoritmo	SG=2	SG=4	SG=8	SG=16	SG=32	SG=64
		cortes	cortes	cortes	cortes	cortes	cortes
memplus	PMSATS	**5333**	**9393**	**11883**	13939	15380	**16761**
	ParMETIS	6143	10511	12703	15348	19636	21338
	GPA	5353	9427	11939	**13279**	**14384**	17409
cs4	PMSATS	361	979	1535	2236	3210	4317
	ParMETIS	435	1207	1877	2704	3654	4915
	GPA	**356**	**936**	**1472**	**2126**	**3080**	**4196**
bcsstk30	PMSATS	**6251**	**16602**	35626	80309	119945	**176287**
	ParMETIS	6676	17376	39360	79481	126567	186779
	GPA	**6251**	16617	**34559**	**70768**	**117232**	177379
bcsstk31	PMSATS	**2676**	8177	14791	26196	41106	**59155**
	ParMETIS	2749	8139	15113	26086	42827	63383
	GPA	**2676**	**7879**	**13561**	**24179**	**38572**	60446
bcsstk32	PMSATS	5049	12417	23456	40627	67475	101501
	ParMETIS	6105	12171	26564	43157	71287	102281
	GPA	**4667**	**9728**	**21307**	**38320**	**62961**	**96168**
t60k	PMSATS	69	**211**	483	889	1473	2322
	ParMETIS	96	247	545	1021	1663	2507
	GPA	**65**	**211**	**467**	**852**	**1420**	**2221**
wing	PMSATS	787	1703	2664	4170	**5980**	8328
	ParMETIS	995	1989	3169	4975	7113	9536
	GPA	**778**	**1636**	**2551**	**4015**	6010	**8161**
brack2	PMSATS	**660**	**2749**	7156	**11858**	18005	**25929**
	ParMETIS	783	3284	7988	13440	20717	29677
	GPA	**660**	2808	**7080**	11958	**17952**	26944
finan512	PMSATS	**162**	**324**	**648**	1620	**2592**	17681
	ParMETIS	**162**	**324**	**648**	**1296**	**2592**	11956
	GPA	**162**	**324**	**648**	**1296**	**2592**	**10821**
fe_tooth	PMSATS	3839	**6942**	**11568**	**17771**	**25528**	**34795**
	ParMETIS	4416	8383	13566	20510	28497	39591
	GPA	**3773**	7152	12646	18435	26016	36030
fe_rotor	PMSATS	**1956**	**7757**	13651	**20674**	**32616**	**46366**
	ParMETIS	2238	8242	14838	23548	36769	52236
	GPA	1957	8097	**13184**	20773	33686	47852
598a	PMSATS	**2336**	8024	**15685**	**25775**	**39098**	**56883**
	ParMETIS	2555	8646	17441	28564	44099	63516
	GPA	**2336**	**7978**	16031	26257	40179	58307
fe_ocean	PMSATS	312	1805	4548	8060	13007	**20709**
	ParMETIS	557	2504	6722	12371	19091	27264
	GPA	**311**	**1704**	**4019**	**7838**	**12746**	21784
wave	PMSATS	8610	**16681**	**29292**	**43029**	**62585**	**84419**
	ParMETIS	9847	19028	34945	48716	69916	94018
	GPA	**8563**	18058	30583	44625	63725	88383
m14b	PMSATS	3842	**13401**	**27468**	**43501**	**66942**	**97143**
	ParMETIS	4219	14607	28689	49184	74007	108724
	GPA	**3802**	13844	27711	44174	68468	101385

Apéndice B

Tablas de resultados (WDND)

Tabla B.1: Resultados obtenidos por trabajos previos en Alperovits-Shamir.

autores	metodología	observaciones	coste
Alperovits y Shamir (1977)	LPG	Subdivisión Tub.	497525
Quindry y col, (1979)	LPG	Subdivisión Tub.	441522
Goulter y col. (1986)	LPG/NS	Subdivisión Tub.	435015
Varma y col. (1986)	NLP (SQP)	Subdivisión Tub.	441310
Fujiwara y col. (1987)	LPG	Subdivisión Tub.	415271
Kessler y Shamir (1989)	LPG	Subdivisión Tub.	417500
Loganathan y col. (1990)	LPG	Subdivisión Tub.	412931
Gupta y col. (1993)	IPF/DPF	Subdivisión Tub.	407625
Eigeret y col. (1994)	LP/BT	Subdivisión Tub.	402352
Loganathan y col. (1995)	MS/SA	Subdivisión Tub.	403561
Sherali y col. (2001)	RLT	Subdivisión Tub.	403385
Savic y Walters (1997)	GA	Discreto	419000
Abebe y Solomatine (1998)	GA	Discreto	424000
Abebe y Solomatine (1998)	CRS2	Discreto	422000
Abebe y Solomatine (1998)	ACCOL	Discreto	447000
Abebe y Solomatine (1998)	CRS4	Discreto	439000
Cunha y Sousa (1999)	SA	Discreto	419000
Zong Woo (2000)	HS	Discreto	419000
Eusuff y Lansey (2003)	SFLA	Discreto	419000
Matias (2003)	GA	Discreto	419000
MENOME (2006)	GA/SA/MSATS/ILSSA/SSSA	Discreto	419000

Tabla B.2: Resultados obtenidos por trabajos previos en Hanoi.

autores	metodología	observaciones	coste
Fujiwara y Khang (1991)	NLPG	Discreto	6145732
Abebe y Solomatine (1998)	GA	Discreto	7000600
Abebe y Solomatine (1998)	ACCOL	Discreto	7836000
Cungha y Sousa (1999)	SA	Discreto	6056000
Vairavamoorthy y Ali (2000)	GA	Discreto	6056370
Zong Woo (2000)	HS	Discreto	6056000
Sherali y col. (2001)	RLT	Subdivisión Tub.	6055542
Eusuff y Lansey (2003)	SFLA	Discreto	6073000
MENOME (2006)	GA	Discreto	6173421

Tabla B.3: Cobertura de conjuntos (SC) de SPEA2 en Alperovits y Shamir.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
B	0.500	—	0.042	0.045	0.120	0.038	0.049	0.069	0.022	0.000	0.098	0.145
C	0.500	0.364	—	0.136	0.400	0.231	0.317	0.172	0.089	0.094	0.256	0.158
D	0.500	0.091	0.208	—	0.160	0.115	0.195	0.103	0.178	0.219	0.197	0.128
E	0.500	0.364	0.125	0.182	—	0.231	0.244	0.069	0.156	0.156	0.225	0.138
F	0.500	0.364	0.125	0.409	0.120	—	0.146	0.069	0.044	0.031	0.201	0.170
G	0.500	0.455	0.333	0.545	0.200	0.231	—	0.207	0.178	0.156	0.312	0.170
H	0.500	0.455	0.292	0.500	0.440	0.192	0.317	—	0.089	0.031	0.313	0.191
I	0.500	0.364	0.292	0.455	0.400	0.308	0.366	0.207	—	0.156	**0.339**	0.154
J	0.500	0.545	0.042	0.273	0.400	0.346	0.293	0.172	0.156	—	0.303	0.167

Tabla B.4: Cobertura de conjuntos (SC) de PESA en Alperovits-Shamir.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.091	0.000	0.100	0.111	0.154	0.000	0.111	0.091	0.000	0.073	0.059
B	0.091	—	0.000	0.200	0.333	0.077	0.000	0.000	0.091	0.182	0.108	0.106
C	0.182	0.000	—	0.200	0.222	0.077	0.000	0.111	0.182	0.091	0.118	0.079
D	0.091	0.182	0.200	—	0.222	0.308	0.375	0.111	0.182	0.091	0.196	0.112
E	0.091	0.091	0.100	0.300	—	0.000	0.125	0.000	0.000	0.091	0.089	0.093
F	0.182	0.182	0.200	0.200	0.111	—	0.125	0.111	0.273	0.273	0.184	0.084
G	0.091	0.182	0.200	0.300	0.222	0.154	—	0.111	0.364	0.273	**0.211**	0.109
H	0.091	0.182	0.100	0.200	0.111	0.154	0.125	—	0.182	0.091	0.137	0.059
I	0.091	0.182	0.200	0.100	0.444	0.154	0.125	0.000	—	0.182	0.164	0.127
J	0.091	0.000	0.200	0.200	0.111	0.308	0.125	0.000	0.273	—	0.145	0.103

Tabla B.5: Cobertura de conjuntos (SC) de PAES en Alperovits-Shamir.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.250	0.000	0.214	0.000	0.000	0.000	0.022	0.031	0.133	0.072	0.078
B	0.167	—	0.000	0.000	0.000	0.000	0.000	0.022	0.000	0.067	0.028	0.053
C	0.750	0.500	—	0.500	0.000	0.133	0.077	0.022	0.031	0.467	0.276	0.268
D	0.250	0.750	0.333	—	0.000	0.033	0.000	0.022	0.000	0.067	0.162	0.144
E	0.667	0.500	0.476	0.571	—	0.367	0.615	0.196	0.188	0.400	0.442	0.218
F	0.250	0.500	0.381	0.571	0.281	—	0.615	0.130	0.406	0.267	0.378	0.193
G	0.750	0.750	0.571	0.571	0.031	0.167	—	0.022	0.156	0.867	0.432	0.332
H	0.583	0.250	0.429	0.500	0.469	0.567	0.654	—	0.594	0.400	**0.494**	0.185
I	0.333	0.750	0.381	0.214	0.406	0.167	0.462	0.087	—	0.067	0.319	0.178
J	0.083	0.250	0.048	0.214	0.000	0.067	0.000	0.000	0.031	—	0.077	0.071

Tabla B.6: Cobertura de conjuntos (SC) de PSA en Alperovits-Shamir.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.000	0.154	0.100	0.000	0.071	0.000	0.091	0.000	0.167	0.065	0.064
B	0.250	—	0.231	0.100	0.077	0.071	0.250	0.091	0.231	0.083	0.154	0.078
C	0.000	0.077	—	0.000	0.000	0.143	0.167	0.000	0.000	0.333	0.080	0.113
D	0.000	0.231	0.231	—	0.077	0.071	0.250	0.000	0.154	0.250	0.140	0.106
E	0.333	0.154	0.231	0.200	—	0.000	0.167	0.273	0.154	0.000	**0.168**	0.119
F	0.167	0.231	0.231	0.200	0.154	—	0.167	0.182	0.077	0.083	0.166	0.073
G	0.083	0.154	0.077	0.100	0.154	0.143	—	0.091	0.231	0.333	0.152	0.094
H	0.000	0.000	0.154	0.000	0.000	0.071	0.000	—	0.077	0.250	0.061	0.084
I	0.083	0.077	0.154	0.100	0.154	0.000	0.083	0.091	—	0.000	0.082	0.059
J	0.167	0.000	0.154	0.100	0.000	0.000	0.167	0.091	0.154	—	0.092	0.070

Tabla B.7: Cobertura de conjuntos (SC) de MOSATS en Alperovits-Shamir.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.000	0.000	0.000	0.154	0.000	0.000	0.000	0.000	0.250	0.045	0.087
B	0.273	—	0.000	0.000	0.231	0.182	0.000	0.000	0.063	0.250	0.111	0.114
C	0.182	0.100	—	0.188	0.231	0.273	0.214	0.071	0.125	0.333	0.191	0.097
D	0.273	0.200	0.167	—	0.385	0.273	0.214	0.000	0.125	0.500	**0.237**	0.158
E	0.091	0.000	0.083	0.063	—	0.000	0.071	0.000	0.063	0.167	0.060	0.051
F	0.273	0.000	0.083	0.250	0.231	—	0.071	0.000	0.188	0.333	0.159	0.116
G	0.091	0.000	0.083	0.000	0.231	0.091	—	0.000	0.125	0.167	0.088	0.075
H	0.000	0.000	0.000	0.000	0.000	0.000	0.000	—	0.000	0.000	0.000	0.000
I	0.364	0.100	0.333	0.250	0.231	0.091	0.143	0.071	—	0.333	0.213	0.124
J	0.000	0.000	0.000	0.000	0.077	0.000	0.000	0.000	0.000	—	0.009	0.024

Tabla B.8: Cobertura de conjuntos (SC) de MOSSSA en Alperovits-Shamir.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.000	0.182	0.063	0.091	0.154	0.167	0.000	0.154	0.091	0.100	0.065
B	0.231	—	0.273	0.313	0.091	0.231	0.167	0.167	0.231	0.182	**0.209**	0.062
C	0.077	0.000	—	0.313	0.000	0.077	0.000	0.000	0.308	0.000	0.086	0.124
D	0.077	0.067	0.000	—	0.000	0.077	0.000	0.000	0.077	0.000	0.033	0.037
E	0.000	0.067	0.091	0.313	—	0.231	0.083	0.000	0.231	0.091	0.123	0.110
F	0.077	0.000	0.091	0.125	0.000	—	0.000	0.000	0.154	0.000	0.050	0.059
G	0.000	0.067	0.091	0.250	0.000	0.231	—	0.000	0.385	0.182	0.134	0.134
H	0.077	0.067	0.000	0.188	0.000	0.154	0.083	—	0.231	0.000	0.089	0.086
I	0.077	0.000	0.000	0.000	0.000	0.000	0.000	0.000	—	0.000	0.009	0.024
J	0.077	0.067	0.182	0.250	0.182	0.154	0.083	0.000	0.385	—	0.153	0.117

Tabla B.9: Cobertura de conjuntos (SC) de SPEA2 en Hanoi.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
B	1.000	—	0.255	0.263	0.140	0.226	0.213	0.175	0.096	0.145	0.279	0.260
C	1.000	0.588	—	0.386	0.386	0.242	0.279	0.238	0.135	0.236	0.388	0.272
D	1.000	0.569	0.235	—	0.246	0.145	0.164	0.159	0.154	0.145	0.313	0.280
E	1.000	0.529	0.314	0.474	—	0.226	0.180	0.222	0.096	0.200	0.360	0.283
F	1.000	0.706	0.294	0.509	0.386	—	0.328	0.317	0.212	0.364	0.457	0.267
G	1.000	0.490	0.392	0.456	0.404	0.435	—	0.381	0.192	0.236	0.443	0.263
H	1.000	0.549	0.412	0.474	0.474	0.258	0.246	—	0.173	0.291	0.431	0.272
I	1.000	0.510	0.529	0.526	0.561	0.290	0.279	0.381	—	0.273	0.483	0.268
J	1.000	0.569	0.412	0.544	0.526	0.339	0.361	0.444	0.308	—	**0.500**	0.257

Tabla B.10: Cobertura de conjuntos (SC) de PESA en Hanoi.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.105	0.077	0.091	0.133	0.118	0.364	0.077	0.071	0.308	0.149	0.113
B	0.714	—	0.077	0.273	0.200	0.706	0.727	0.231	0.500	0.769	0.466	0.257
C	0.857	0.842	—	0.545	0.533	0.824	0.636	0.462	0.500	0.769	**0.663**	0.261
D	0.643	0.316	0.077	—	0.467	0.529	0.636	0.077	0.214	0.538	0.389	0.247
E	0.571	0.053	0.077	0.000	—	0.294	0.455	0.154	0.214	0.385	0.245	0.191
F	0.571	0.105	0.077	0.091	0.200	—	0.636	0.077	0.143	0.538	0.271	0.235
G	0.143	0.000	0.000	0.000	0.067	0.000	—	0.077	0.000	0.077	0.040	0.049
H	0.500	0.211	0.077	0.182	0.333	0.412	0.545	—	0.071	0.231	0.285	0.187
I	0.714	0.263	0.308	0.455	0.667	0.588	0.727	0.462	—	0.538	0.525	0.219
J	0.214	0.000	0.000	0.091	0.000	0.176	0.364	0.77	0.000	—	0.102	0.120

Tabla B.11: Cobertura de conjuntos (SC) de PAES en Hanoi.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.500	0.000	0.105	0.000	0.375	0.000	0.000	0.250	0.200	0.159	0.144
B	0.000	—	0.000	0.105	0.000	0.375	0.000	0.000	0.000	0.067	0.061	0.117
C	0.813	0.500	—	0.105	0.333	0.875	0.087	0.000	0.750	0.267	0.414	0.343
D	0.813	0.500	0.214	—	0.556	0.500	0.043	0.133	0.250	0.400	0.379	0.257
E	0.813	0.500	0.214	0.105	—	0.750	0.087	0.067	0.250	0.267	0.339	0.285
F	0.063	0.000	0.000	0.105	0.000	—	0.000	0.000	0.000	0.133	0.033	0.050
G	0.875	0.750	0.250	0.842	0.778	0.750	—	0.200	0.500	0.467	0.601	0.309
H	0.875	0.750	0.857	0.789	0.778	0.875	0.217	—	0.750	0.400	**0.699**	0.319
I	0.063	0.000	0.000	0.053	0.000	0.125	0.000	0.000	—	0.000	0.027	0.042
J	0.188	0.500	0.143	0.211	0.556	0.500	0.174	0.167	0.250	—	0.299	0.175

Tabla B.12: Cobertura de conjuntos (SC) de PSA en Hanoi.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.167	0.286	0.083	0.250	0.182	0.143	0.133	0.286	0.500	0.225	0.138
B	0.077	—	0.357	0.083	0.250	0.091	0.071	0.133	0.286	0.313	0.185	0.109
C	0.154	0.333	—	0.083	0.375	0.182	0.143	0.067	0.286	0.313	0.215	0.123
D	0.077	0.250	0.143	—	0.125	0.091	0.143	0.133	0.214	0.250	0.158	0.075
E	0.077	0.333	0.143	0.083	—	0.000	0.286	0.333	0.357	0.375	0.221	0.150
F	0.154	0.250	0.429	0.167	0.375	—	0.214	0.133	0.214	0.313	0.250	0.127
G	0.154	0.167	0.429	0.333	0.250	0.273	—	0.200	0.214	0.313	0.259	0.116
H	0.385	0.333	0.286	0.333	0.250	0.091	0.367	—	0.500	0.438	**0.330**	0.156
I	0.385	0.333	0.500	0.333	0.438	0.091	0.286	0.133	—	0.375	0.319	0.165
J	0.077	0.167	0.286	0.083	0.188	0.182	0.143	0.267	0.214	—	0.178	0.090

Tabla B.13: Cobertura de conjuntos (SC) de MOSATS en Hanoi.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.333	0.154	0.357	0.462	0.400	0.267	0.545	0.214	0.643	0.375	0.193
B	0.077	—	0.154	0.214	0.385	0.467	0.067	0.182	0.286	0.429	0.251	0.140
C	0.231	0.250	—	0.429	0.538	0.667	0.200	0.455	0.500	0.571	**0.427**	0.202
D	0.154	0.167	0.077	—	0.462	0.533	0.133	0.273	0.571	0.357	0.303	0.197
E	0.154	0.167	0.231	0.214	—	0.333	0.267	0.182	0.357	0.286	0.243	0.103
F	0.231	0.250	0.077	0.286	0.231	—	0.200	0.273	0.071	0.214	0.204	0.099
G	0.154	0.333	0.231	0.286	0.462	0.533	—	0.364	0.500	0.571	0.382	0.185
H	0.154	0.250	0.077	0.143	0.231	0.333	0.133	—	0.357	0.571	0.250	0.166
I	0.077	0.083	0.154	0.214	0.385	0.333	0.200	0.182	—	0.286	0.213	0.114
J	0.077	0.083	0.154	0.143	0.231	0.400	0.067	0.091	0.214	—	0.162	0.112

Tabla B.14: Cobertura de conjuntos (SC) de MOSSSA en Hanoi.

	A	B	C	D	E	F	G	H	I	J	AVG	D.T.
A	—	0.308	0.529	0.462	0.333	0.563	0.286	0.300	0.500	0.643	0.436	0.187
B	0.250	—	0.353	0.462	0.533	0.500	0.500	0.400	0.500	0.714	**0.468**	0.122
C	0.125	0.308	—	0.692	0.400	0.438	0.357	0.300	0.600	0.714	0.437	0.232
D	0.063	0.077	0.118	—	0.133	0.250	0.214	0.100	0.500	0.571	0.225	0.185
E	0.250	0.231	0.235	0.616	—	0.438	0.357	0.200	0.500	0.786	0.401	0.226
F	0.188	0.231	0.235	0.231	0.333	—	0.286	0.200	0.600	0.786	0.343	0.224
G	0.125	0.231	0.235	0.462	0.267	0.500	—	0.300	0.500	0.714	0.370	0.209
H	0.250	0.077	0.235	0.615	0.533	0.438	0.429	—	0.500	0.571	0.405	0.187
I	0.000	0.000	0.059	0.000	0.200	0.000	0.143	0.000	—	0.500	0.100	0.158
J	0.063	0.077	0.176	0.077	0.067	0.063	0.143	0.100	0.100	—	0.096	0.049

Bibliografía

Bibliografía

[ABE02] Ababei, C., Selvakkumaran, N., Bazargan, K., Karypis, G., *Multiobjective Circuit Partitioning for Cutsize and Path-based Delay Minimization*, In Proceedings of the IEEE/ACM International Conference on Computer Aided Design, 2002, pp. 181-185.

[AGU] *American Geophysical Union*, http://www.agu.org/

[ALB02] Alba, E., Tomassini M., *Parallelism and Evolutionary Algorithms*, IEEE Transactions on Evolutionary Computation 6(5), 2002, pp. 443-462.

[ALE01] Aleta, A., Codina, J.M., Sánchez, J., González, A., *Graph-Partitioning Based Instruction Scheduling for Clustered Processors*, In Proceedings 34th ACM/IEEE Interational Symposium on Microarchitecture, 2001, pp 150-159.

[ALP77] Alperovits, E., Shamir, U., *Design of Optimal Water Distribution Systems*, Water Resources Research 13(6), 1977, pp. 885-900.

[ALT95] Alpert, C.J., Kahng, A., *Recent Developments in Netlist Partitioning: A Survey*, Integration, the VLSI Journal 19(1-2), 1995, pp. 1-81.

[AMD67] Amdahl, G.M., *Validity of the Single-processor Approach to Achieving Large-scale Computing Capabilities*, In Proceedings of the American Federation of Information Processing Societies 30, Washington, 1967, pp. 483-485.

[BAÑ03] Baños R., Gil, C., Ortega, J., Montoya, F.G., *Multilevel Heuristic Algorithm for Graph Partitioning*, Springer-Verlag Lecture Notes in Computer Science 2611, 2003, pp. 143-153.

[BAÑ04a] Baños R., Gil, C., Ortega, J., Montoya, F.G., *Optimising Graphs Partitions using Parallel Evolution*, Springer-Verlag Lecture Notes in Computer Science 2936, 2004, pp. 91-102.

[BAÑ04b] Baños R., Gil, C., Ortega, J., Montoya, F.G., *A Parallel Multilevel Metaheuristic for Graph Partitioning*, Journal of Heuristics 10(3), 2004, pp. 315-336.

[BAÑ06a] Baños R., Gil, C., Paechter, B., Ortega, J., *A Hybrid Meta-heuristic for Multi-objective Optimization: MOSATS*, Journal of Mathematical Modelling and Algorithms, DOI: 10.1007/s10852-006-9041-6, 2006.

[BAÑ06b] Baños R., Gil, C., Paechter, B., Ortega, J., *Parallelization of Population-based Multi-Objective Metaheuristics: An Empirical Study*, Applied Mathematical Modelling 30(7), 2006, pp. 578-592.

[BAÑ06c] Baños R., Gil, C., Montoya, M.G., Ortega, J., *Adapting Multi-Objective Meta-Heuristics for Graph Partitioning*, In Applied Soft Computing Technologies: The Challenge of Complexity, Series: Advances in Soft Computing, Springer, 2006, pp. 123-132.

[BAÑ06d] Baños R., Gil, C., Reca, J., Martínez, J., *Constrained Optimization of Water Distribution Networks using Meta-heuristics*, Enviado a Constrains Journal, 2006.

[BAÑ06e] Baños R., Gil, C., Reca, J., Martínez, J., *Implementation of Scatter Search for Multi-Objective Optimization: A Comparative Study*, Computational Optimization and Applications J., (aceptado).

[BAÑ06f] Baños R., Gil, C., *Graph-Mesh Partitioning: An Overview of the Current State-of-the-art*, In Mesh Partitioning Tecnhiques and Domain Decomposition Methods, F. Magoules (ed.), Saxe-Coburg Publications (en prensa).

[BAR03] Barichard, V., Hao J.K, *Genetic Tabu Search for the Multi-objective Knapsack Problem*, Journal of Tsinghua Science and Technology 8(1), 2003, pp. 8-13.

[BAS95] Barnard, S.T., Simon, H.D., *A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems*, Concurrency: Practice and Experience 6(2), 1994, pp. 101-117.

[BAT97] Battiti, R., Bertossi, A.A., *Differential Greedy for the 0-1 Equicut Problem*, In Network Design: Connectivity and Facilities, D. Zu y P.M. Pardalos (eds.), Princeton: American Mathematical Society, vol. 40, 1997, pp. 3-22.

[BEA06] Beausoleil, R., *MOSS - Multiobjective Scatter Search Applied to Nonlinear Multiple Criteria Optimization*, European Journal of Operational Research 169(2), 2006, pp. 426-449.

[BEE73] Berge, C., *Graphs and Hypergraphs*, North-Holland, Amsterdan, 1973.

[BEG87] Berger, M.J., Bokhari, S.H., *A Partitioning Strategy for Nonuniform Problems on Multiprocessors*, IEEE Transactions on Computers 36(5), 1987, pp. 570-580.

[BEK02] Berkhin, P., *Survey of Clustering Data-mining Techniques*, Accrue Software, San Jose, USA, 2002.

[BEN00] Bern, M., Plassmann, P., *Mesh Generation*, Handbook of Computational Geometry, J.R. Sack y J. Urrutia (eds.), Elsevier Science, 2000.

[BER95] Berry, J., Goldberg, M., *Path Optimization and Near-greedy Analysis for Graph Partitioning: An Empirical Study*, In Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms, 1995, pp. 223-232.

[BLA00] Blazewicz, J., *Handbook on Parallel and Distributed Processing*, In International Handbooks on Information Systems, Springer, 2000.

[BLU03] Blum C., Roli, A., *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison*, ACM Computing Surveys 35(3), 2003, pp. 268-308.

[BOO82] Booker, L.B., *Intelligent Behavior as an Adaptation to the Task Environment*, Doctoral dissertation, Technical Report No. 243, Ann Arbor: University of Michigan, Logic of Computers Group), Dissertation Abstracts International 43(2), 469B.

[BUK05] Burk, F., *Lebesgue Measure And Integration: An Introduction*, John Wiley & Sons Inc, 2005.

[BUR01] Burke, E.K., Cowling, P., Landa Silva, J.D., Petrovic S., *Combining Hybrid Metaheuristics and Populations for the Multiobjective Optimisation of Space Allocation Problems*, In Proceedings of the 2001 GECCO, San Francisco USA, 2001, pp. 1252–1259.

[CAH04] Cahon, S., Melab, N., Talbi, E-G., *ParadisEO: a Framework for the Flexible Design of Parallel and Distributed Hybrid Metaheuristics*, Journal of Heuristics 10(3), 2004, pp. 357-380.

[CAN97] Cantu-Paz, E., *A Survey of Parallel Genetic Algorithms*, Technical Report IlliGAL 97003, University of Illinois at Urbana-Champaign, 1997.

[CAT99] Catalyurek, U., Aykanat, C., *Hypergraph-partitioning-based Decomposition for Parallel Sparse-matrix Vector Multiplication*, IEEE Transactions on Parallel and Distributed Computing 10(7), 1999, pp. 673-693.

[CEN97] Censor, Y., Zenios, S.A., *Parallel Optimization, Theory, Algorithms and Applications*, Oxford University Press, Oxford (1997).

[CHA94] Hendrickson, B., Leland, R., *Chaco v. 2.0, Technical Report, SAND94-2692.* http://www.cs.sandia.gov/∼bahendr/chaco.html

[CHE04] Chen, H., Zeng, Q., Agrawal, D., *A Novel Optimal Channel Partitioning Algorithm for Integrated Wireless and Mobile Networks*, Wireless Networks 10(5), 2004, pp. 507-517.

[COE98] Coello Coello, C.A., *An Updated Survey of GA-Based Multiobjective Optimization Techniques*, Technical Report Lania-RD-98-08, Laboratorio Nacional de Informática Avanzada (LANIA), Mexico, 1998.

[COE00] Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B., *Evolutionary Algorithms for Solving Multi-objective Problems*, Kluwer Academic Publishers, New York, 2002.

[COM06] Comellas, F., Sapena, E., *A Multiagent Algorithm for Graph Partitioning*, Springer-Verlag Lecture Notes in Computer Science 3907, 2006, pp. 279-285.

[COO71] Cook, S., *The Complexity of Theorem Proving Procedures*, In Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, pp. 151158.

[COR00] Corne D.W., Knowles J.D., Oates, M.J., *The Pareto-envelope Based Selection Algorithm for Multiobjective Optimization*, Springer-Verlag Lecture Notes in Computer Science 1917, 2000, pp. 839-848.

[COR01] Corne D.W., Jerran, N.R., Knowles J.D., Oates, M.J., *PESA-II: Region-based Selection in Evolutionary Multi-objective Optimization*, In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), Morgan Kaufmann Publishers, 2001, pp. 283–290.

[CUL98] Culler, D., Pal-Singh, J., Gupta, A., *Parallel Computer Architecture. A Hardware/Software Approach*, In Morgan Kaufmann Series in Computer Architecture and Design, 1st edition, 1998.

[CUN01] Cung, V.D., Martins, S.L., Ribeiro, C.C., Roucairol, C., *Strategies for the Parallel Implementation of Metaheuristics*, In Essays and Surveys in Metaheuristics, C. Ribeiro y P. Hansen (eds.), Kluwer, 2001, pp. 263-308.

[CUS99] Cunha M.D., Sousa, J., *Water Distribution Network Design Optimization: Simulated Annealing Approach*, Water Resources Planning and Management (ASCE) 125(4), 1999, pp. 215-221.

[CZY98] Czyzak, P., Jaszkiewicz A., *Pareto Simulated Annealing - a Metaheuristic Technique for Multiple-objective Combinatorial Optimization*, Journal of Multi-Criteria Decision Analysis 7(1), 1998, pp. 34-47.

[DEB00] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., *A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multiobjective Optimization: NSGA-II*, Springer-Verlag Lecture Notes in Computer Science 1917, 2000, pp. 849-858.

[DEB01] Deb, K., *Multi-objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, New York, 2001.

[DEK91] Deekers, A., Aarts, E., *Global Optimization and Simulated Annealing*, Mathematical Programming 50, 1991, pp. 367-393.

[DEN90] Deneubourg, J.L., Aron, S., Goss, S., Pasteels, J.M. *The Self-organizing Exploratory Patterns of the Argentine Ant*, Journal of Insect Behaviour 3, 1990, pp. 159-168.

[DEV05] Devine, K.D., Boman, E.G., Heaphy, R.T., Hendrickson, B., Teresco, J.D., Faik, J., Flaherty, J.E., Gervasio, L.G., *New Challenges in Dynamic Load Balancing*, Applied Numerical Mathematics 52(2-3), 2005, pp. 133-152.

[DIE96] Diekmann, R., Luling, R., Monien, B., Spraner, C., *Combining Helpful Sets and Parallel Simulated Annealing for the Graph-Partitioning Problem*, Parallel Algorithms and Applications 8, 1996, pp. 61-84.

[DOR99] Dorigo M., Di Caro, G., *The Ant Colony Optimization Meta-heuristic*, In New Ideas in Optimization, F. Glover (ed.), McGraw-Hill, 1999, pp. 11-32.

[EIG94] Eiger, G., Shamir, U., Ben-Tal, A., *Optimal Design of Water Distribution Networks*, Water Resources Research 30(9), 1994, pp. 2637-2646.

[EMO] *List of References on Evolutionary Multiobjective Optimization*, http://www.lania.mx/

[ELS02] Elsner, U., *Static and Dynamic Graph Partitioning. A Comparative Study of Existing Algorithms*, Logos Verlag, Berlin, 2002.

[ERC05] Erciyes, K., Soysert, Z., *Multilevel Static Real-time Scheduling Algorithms using Graph Partitioning*, Springer-Verlag Lecture Notes in Computer Science 3514, 2005, pp. 196-203.

[EUS03] Eusuff, M.M., Lansey, K.E., *Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm*, Journal of Water Resources Planning and Management (ASCE) 129(3), 2003, pp. 210-225.

[FAH88] Farhat, C., *A Simple and Efficient Automatic FEM Domain Decomposer*, Computer and Structures 28(5), 1988, pp. 579-602.

[FAR03] Farmani, R., Savic, D.A., Walters, G.A., *Multi-objective Optimization of Water System: A Comparative Study*, In Pumps, Electromechanical Devices and Systems Applied to Urban Water Management, E. Cabrera y col. (eds.), Vol. 1, Rotterdam, 2003, pp. 247-256.

[FAR05] Farmani, R., Walters, G.A., Savic, D.A., *Trade-off Between Total Cost and Reliability for Anytown Water Distribution Network*, Journal of Water Resources Planning and Management (ASCE) 131(3), 2005, pp. 161-171.

[FEO95] Feo, T., Resende, M., *Greedy Randomized Adaptive Search Procedures*, Journal of Global Optimization 6, 1995, pp. 109-133.

[FID82] Fiduccia, C., Mattheyses, R., *A Linear Time Heuristic for Improving Network Partitions*, In Proceedings of the 19th IEEE Design Automation Conference, 1982, pp. 175-181.

[FOG06] Fogel, D.B., *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, Wiley-IEEE Press, 2006.

[FON93] Fonseca C.M., Fleming P.J., *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization*, In Proceedings of the 5th International Conference on Genetic Algorithms, 1993, pp. 416-423.

[FUJ90] Fujiwara, O., Khang, D.B., *A Two-phase Decomposition Method for Optimal Design of Looped Water Distribution Networks*, Water Resources Research 26(4), 1990, pp. 539-549.

[GAR79] Garey M.R., Johnson D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman & Company, San Francisco, 1979.

[GEE01] Geem, Z.W., Kim J.H., Loganathan G.V., *A New Heuristic Optimization Algorithm: Harmony Search*, Simulation 76(2), 2001, pp. 60-68.

[GEI94] Geist A., Beguelin, A., Dongarra J., Jiang W., Manchek, R., *PVM3 User's Guide and Reference Manual*, Oak Ridge Natinal Laboratory, Oak Ridge, Tennessee, 1994.

[GEN01] Gendreau, M., Laporte, G., Semet, F., *A Dynamic Model and Parallel Tabu Search Heuristic for Real-time Ambulance Relocation*, Parallel Computing 27(12), 2001, pp. 1641-1653.

[GIL96] Gil, C., *Procedimientos Paralelos Basados en los Coeficientes de Reed-Muller para la Generación de Patrones de Test en Circuitos Digitales*, Tesis Doctoral, Universidad de Granada, 1996.

[GIL98] Gil, C., Ortega, J., Díaz, A.F., Montoya, M.G., *Annealing-based Heuristics and Genetic Algorithms for Circuit Partitioning in Parallel Test Generation*, Future Generation Computer Systems 14(5), 1998, pp. 439-451.

[GIL02] Gil, C., Ortega, J., Montoya, M.G., Baños R., *A Mixed Heuristic for Circuit Partitioning*, Computational Optimization and Applications Journal 23(3), 2002, pp. 321-340.

[GLB98] Gilbert, J., Miller, G., Teng, S., *Geometric Mesh Partitioning: Implementation and Experiments*, SIAM Journal on Scientific Computing 19(6), 1998, pp. 2091-2110.

[GLO77] Glover, F., *Heuristics for Integer Programming using Surrogate Constraints*, Decision Sciences 8(1), 1997, pp. 555-568.

[GLO93] Glover, F., Laguna, M., Dowsland K.A., *Modern Heuristic Techniques for Combinatorial Problems*. C.R. Reeves (ed.), Blackwell, London, 1993.

[GLO97] Glover, F., Laguna, M., *Tabu search*. Kluwer Academic Publishers, Boston, 1997.

[GLO01] Glover, F., Gutin, G., Yeo A., Zverovich, A., *Construction Heuristics for the Asymmetric TSP*, European Journal of Operational Research 129, 2001, pp. 555-568.

[GLO03a] Glover, F., Kochenberger, G.A., *Handbook of Meta-heuristics*, International Series in Operations Research Management Science, 57.

[GLO03b] Glover, F., Melián, B., *Búsqueda Tabu*, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 19(2), 2003, pp. 7-28.

[GOL89] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, New York, 1989.

[GOL91] Goldberg, D.E., Deb, K., *A Comparison of Selection Schemes used in Genetic Algorithms*, In Foundations of Genetic Algorithms, G.J.E. Rawlings (ed.), Addison-Wesley, Redwood City, CA, 1991, pp. 69-93.

[GOR99] Gordon, D.B., Mayo S.L., *Branch-and-terminate: A Combinatorial Optimization Algorithm for Protein Design*, Structure 7(9), 1999, pp. 1089-1098.

[GPA] Graph Partitioning Archive,
http://staffweb.cms.gre.ac.uk/∼c.walshaw/partition/

[GRE84] Grefenstette, J.J., *GENESIS: A System for Using Genetic Search Procedures*, In Proceedings of the Conference on Intelligent Systems and Machines, 1984, pp. 161-165.

[GRO95] Grotschel, M., *Combinatorial Optimization*, In Handbook of Combinatorics, vol. 2, chapter 28, R. Graham y col. (eds.), North–Holland, 1995, pp. 1541-1597.

[GUP93] Gupta, I., Bassin, J.K., Gupta A., Khanna P., *Optimization of Water Distribution System*, Environmental Software 8, 1993, pp. 101-113.

[HAG91] Hagersten E., Landin A., Haridi, S., *Multiprocessor Consistency and Synchronization Through Transient Cache States*, Kluwer Academic Publishers, Boston MA, 1991.

[HAJ92] Hajela P., Y-Lin C., *Genetic Search Strategies in Multi-criterion Optimal Design*, Structural Optimization 4, 1992, pp. 99–107.

[HAN97] Hansen, P., *Tabu Search for Multiobjective Optimization: MOTS*, In Proceedings of the 13th International Conference on Multiple Criteria Decision Making, 1997.

[HAN99] Hansen, P., Mladenovic, N., *An Introduction to Variable Neighborhood Search*, In Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization, S. Voss (ed.), Kluwer Academic Publishers, Dordrecht, 1999, pp. 433-458.

[HAS94] Hasan, H., *Space Filling Curves*, Springer, 2004.

[HAW98] Hwang K., Xu Z., *Scalable Parallel Computers*, McGraw-Hill, 1998.

[HEN93] Hendrickson, B., Leland, R., *A Multilevel Algorithm for Partitioning Graphs*, Technical Report SAND93-1301, Sandia National Laboratories, 1993.

[HEN95] Hendrickson, B., Leland, R., *An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations*, SIAM Journal on Scientific Computing 16(2), 1995, pp. 452-469.

[HEN00] Hendrickson, B., Devine, K., *Dynamic Load Balancing in Computational Mechanics*, Computer Methods in Applied Mechanics and Engineering 184(2-4), 2000, pp. 485-500.

[HES02] Henessy J., Patterson D., *Computer Architecture: A Cuantitative Approach*, In Morgan Kaufmann Series in Computer Architecture and Design, 3rd edition, 2002.

[HOL75] Holland, J., *Adaptation in Natural and Artificial Systems*, MIT Press, 1975.

[HOR94] Horn, J., Nafpliotis, N., Goldberg, D., *A Niched Pareto Genetic Algorithm for Multiobjective Optimization*, In Proceedings of the 1st IEEE Conference on Evolutionary Computation, Piscataway, New Jersey, USA, 1994, pp. 82–87.

[HOS95] Horst, R., Pardalos, P.M., *Handbook of Global Optimization*, Kluwer, Dordrecht, 1995.

[HU03] Hu, X., Huang, Z., Wang, Z., *Hybridization of the Multi-objective Evolutionary Algorithms and the Gradient-based Algorithms*, In Proceedings of the Congress on Evolutionary Computation, 2003, pp. 870-877.

[ISH98] Ishibuchi, H. Murata, T., *A Multi-objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling*, IEEE Transactions on Systems, Man, and Cybernetics- Part C: Applications and Reviews 28(3), 1998, pp. 392-403.

[JAC04] Jacobson, S.H., Yucesan, E., *Analyzing the Performance of Generalized Hill Climbing Algorithms*, Journal of Heuristics 10(4), 2004, pp. 387-405.

[JIA03] Jiang, X., Shen, X., Zhang, T., Liu, H., *An Improved Circuit-partitioning Algorithm based on Min-cut Equivalence Relation*, Integration, the VLSI Journal 36(1), 2003, pp. 55-68.

[JOH89] Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C., *Optimization by Simulated Annealing: An Experimental Evaluation. Part I, Graph Partitioning*, Operations Research 37(6), 1989, pp. 865-892.

http://staffweb.cms.gre.ac.uk/∼c.walshaw/jostle/

[KAR98a] Karypis, G., Kumar V., *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*, SIAM Journal on Scientific Computing 20(1), 1998, pp. 359-392.

[KAR98b] Karypis, G., Kumar, V., *A Parallel Algorithm for Multilevel Graph Partitioning and Sparse Matrix Ordering*, Journal of Parallel and Distributed Computing 48(1), 1998, pp. 71-95.

[KAR98c] Karypis, G., Kumar, V., *Multilevel k-way Partitioning Scheme for Irregular Graphs*, Journal of Parallel and Distributed Computing 48(1), 1998, pp. 96-129.

[KAR99] Karypis, G., Aggarwal, R., Kumar, V., *Multilevel Hypergraph Partitioning: Applications in VLSI Domain*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 7(1), 1999, pp. 69-79.

[KEL98] Kelly, F.P., Maulloo A.K., Tan, D.K., *Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability*, Journal of the Operational Research Society 49, 1998, pp. 237-252.

[KEN06] Kelner, J.A., *Spectral Partitioning, Eigenvalue Bounds, and Circle Packings for Graphs of Bounded Genus*, SIAM Journal of Computing, 35(4), 2006, pp. 882-902.

[KER70] Kernighan, B.W., Lin, S., *An Efficient Heuristic Procedure for Partitioning Graphics*, The Bell System Technical Journal 1970, pp. 291-307.

[KIR83] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., *Optimization by Simulated Annealing*, Science 220, 1983, pp. 671.

[KNO99] Knowles, J.D., Corne, D.W., *The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation*, In Proceedings of the Congress on Evolutionary Computation, Vol. 1, Piscataway, NJ. IEEE Press, 1999, pp. 98-105.

[KNO00] Knowles, J.D., Corne, D.W., *M-PAES: A Memetic Algorithm for Multiobjective Optimization*, In Proceedings of the Congress on Evolutionary Computation, 2000, Piscataway, NJ. IEEE Press, 2000. pp. 325-332.

[KNO04] Knowles, J.D., Corne, D.W., *Bounded Pareto Archiving: Theory and Practice*, Springer-Verlag Lecture Notes in Economics and Mathematical Systems 535, 2004, pp. 39-64.

[KOR04] Korosec, P., Silc, J., Robic, B., *Solving the Mesh-partitioning Problem with an Ant-colony Algorithm*, Parallel Computing 30(5-6), 2004, pp. 785-801.

[KOY05] Koyoturk, M., Aykanat, C., *Iterative-improvement-based Declustering Heuristics for Multi-disk Databases*, Information Systems 30(1), 2005, pp. 47-70.

[KRI89] Krishnakumar, K., *Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimization*, In Proceedings of the SPIE: Intelligent Control and Adaptive Systems, 1989, vol. 1196, pp. 289-296.

[KUB02] Kubalik, J., Lazansk J., Zikl P., *Layout Problem Optimization Using Genetic Algorithms*, In Proceedings of the 5th IFIP/IEEE International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS'02), 2002, pp. 493-500.

[KUM03] Kumar, R., Rockett, P., *Evolutionary Multimodal Optimization Revisited*, In Proceedings of the Genetic and Evolutionary Computation Conference, 2003, pp. 1592-1593.

[LAR06] Larson, J., *Direct Graph k-Partitioning with a Kernighan-Lin Like Heuristic*, Operations Research Letters 34(6), 2006, pp. 621-629.

[LAU01] Laumanns M., Zitzler E., Thiele L., *On the Effects of Archiving, Elitism, and Density Based Selection in Evolutionary Multi-objective Optimization*, Springer-Verlag Lecture Notes in Computer Science 1993, 2001, pp. 181-196.

[LEE80] Lee, R.B., *Empirical Results on the Speedup, Efficiency, Redundancy, and Quality of Parallel Computations*, In Proceedings of the International Conference on Parallel Processing, 1980, pp. 91-96.

[LIN00] Linares, P., Romero, C., *A Multiple Criteria Decision Making Approach for Electricity Planning in Spain: Economic versus Environmental Objectives*, Journal of the Operational Research Society 51, 2000, pp. 736-743.

[MAH95] Mahfoud, S.W., *Niching Methods for Genetic Algorithms*, PhD thesis, University of Illinois at Urbana-Champaign, 1995. IlliGAL Report No. 95001.

[MAI03] Maier, H.R., Simpson A.R., Zecchin, A.C., Foong W.K., Phang K.Y., Seah H.Y., Tan, C.L., *Ant Colony Optimization for Design of Water Distribution Systems*, Water Resources Planning and Management (ASCE) 129(3), 2003, pp. 200-209.

[MAR06] Marti, R., Laguna, M., Glover, F., *Principles of Scatter Search*, European Journal of Operations Research 169(2), 2006, pp. 359-372.

[MAY00] Mays L.W., *Water Distribution System Handbook*, McGraw-Hill, USA, 2000.

[MED05] Mérida-Casermeiro, E., López-Rodriguez D., *Graph Partitioning Via Recurrent Multivalued Neural Networks*, Springer-Verlag Lecture Notes in Computer Science 3512, 2005, pp. 1149-1156.

[MER00] Merz, P., Freisleben, B., *Fitness Landscapes, Memetic Algorithms and Greedy Operators for Graph Bi-partitioning*, Evolutionary Computation 8(1), 2000, pp. 61-91.

[MES] *METIS - Family of Multilevel Partitioning Algorithms*, Dpt. of Computer Science and Engineering, University of Minnesota, USA. http://www.glaros.dtc.umn.edu/gkhome/views/metis

[MET53] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E. *Equation of State Calculations by Fast Computing Machines*, Journal of Chemical Physics 21(6), 1953, pp. 1087-1092.

[MIL93] Miller, G.L., Teng, S., Thurston, W., Vavasis, S.A., *Automatic Mesh Partitioning*, In Sparse Matrix Computations: Graph Theory Issues and Algorithms, George y col. (eds.), IMA volumes in Mathematics and its Applications, Springer-Verlag pp. 57-84, 1993.

[MOL06] Molina, J., Laguna, M., Martí, R., Caballero, R., *SSPMO: A Scatter Tabu Search Procedure for Non-linear Multiobjective Optimization*, Informs Journal on Computing, (en prensa).

[MON99] Montesinos, P., Garcia-Guzman A., Ayuso J.L., *Water Distribution Network Optimization using Modified Genetic Algorithm*, Water Resources Research 35(11), 1999, pp. 3467-3473.

[MOS00] Moscato, P., *Memetic Algorithms*, In Handbook of Applied Optimization, P.M. Pardalos y M.G.C. Resende (eds.), Oxford University Press, 2000.

[MOT05] Mott R.L., *Applied Fluid Mechanics*, Prentice-Hall, USA, 2005.

[NEB05] Nebro, A.J., Luna, F., Alba, E., *New Ideas in Applying Scatter Search to Multiobjective Optimization*, Springer-Verlag Lecture Notes in Computer Science 3410, 2005, pp. 443-458.

[NOI98] Nourani, Y., Andresen, B., *A Comparison of Simulated Annealing Cooling Strategies*, Journal of Physics A: Mathematical and General 31(41), 1998, pp. 8373-8385.

[NOU87] Nour-Omid, B., Raefsky, A., Lyzenga, G., *Solving Finite Element Equations on Concurrent Computers*, Parallel Computations and Their Impact on Mechanics, A.K. Noor (ed.), ASME, New York, 209-227, 1986.

[OPE98] *OpenMP C and C++ Application Program Interface versión 1.0*, 1998.

[PAR00] Pardalos, P.M., Resende, M.G.C. (eds.), *Handbook of Applied Optimization*, Oxford University Press, 2000.

[POT90] Pothen, A., Simon, H., Liou, K., *Partitioning Sparse Matrices with Eigenvectors of Fraphs*, SIAM Journal of Matrix Analysis and Applications 11(3), 1990, pp. 430-452.

[PUC05] Puchinger, J., Raidl, G., *Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey Classification*, Springer-Verlag Lecture Notes in Computer Science 3562, 2005, pp. 41-53.

[RAM02] Ramalhino, H., Martin, O., Stutzle, T., *Iterated Local Search*, In Handbook of Metaheuristics, F. Glover y G. Kochenberger (eds.), Kluwer Academic Publishers, Norwell, 2001, pp. 321-353.

[RAN99] Randall M., Abramson, A., *A Parallel Tabu Search Algorithm for Combinatorial Optimisation Problems*, In Proceedings of the 6th Australasian Conference on Parallel and Real Time Systems, Singapore, 1999, pp. 68-79.

[RAS03] Rastello, F., Rao, A., Pande, S., *Optimal Task Scheduling to Minimize Inter-Tile Latencies*, Parallel Computing 29(2), 2003, pp. 209-239.

[REC06] Reca, J., Martínez, J., *Genetic Algorithms for the Design of Looped Irrigation Water Distribution Networks*, Water Resources Research 42(5), W05416, doi:10.1029/2005WR004383, 2006.

[RES03] Resende, M., González-Velarde J.L., *GRASP: Procedimientos de Búsqueda Miopes Aleatorizados y Adaptativos*, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 19, 2003, pp. 61-76.

[RIC89] Richardson, T., Palmer, M.R., Liepins, G., Hilliard, M., *ome Guidelines for Genetic Algorithms with Penalty Functions*, In Proceedings of the 3rd International Conference on Genetic Algorithms, Schaffer (ed.), George Mason University, Morgan Kaufman Publishers, 1989, pp. 191-197.

[ROC00] Rockafellar, R.T., Uryasev, S., *Optimization of Conditional Value-at-risk*, Journal of Risk 2(3), 2000, pp. 21-41.

[ROL96] Rolland, E., Pirkul, H.P., Glover, F., *Tabu Search for Graph Partitioning*, Annals of Operations Research 63, 1996, pp. 209-232.

[ROS00] Rossman, L.A., *EPANET 2 User's Manual*, EPA/600/R-00/057, 2000.

[RSB67] Rosenberg, R.S., *Simulation of Genetic Populations with Biochemical Properties*, PhD thesis, University of Michigan, Ann Harbor, Michigan, 1967.

[RUD00] Rudolph, G., Agapie A., *Convergence Properties of Some Multi-Objective Evolutionary Algorithms*, In Proceedings of the 2000 Conference on Evolutionary Computation, Piscataway, New Jersey, vol.2, 2000, pp. 1010–1016.

[RUM02] Rummler, A., Apetrei A., *Graph Partitioning Revised - a Multiobjective Perspective*, In Proceedings of the 6th World Conference on Systemics, Cybernetics and Informatics, Orlando, Florida, USA, 2002.

[SAR98] Sareni, B., Krahenbuhl, L., *Fitness Sharing and Niching Methods Revisited*, IEEE Transactions on Evolutionary Computation 2(3), 1998, pp. 97-106.

[SAV97] Savic, D.A., Walters G.A., *Genetic Algorithms for Least-cost Design of Water Distribution Networks*, Water Resources Planning and Management (ASCE) 123(2), 1997, pp. 67-77.

[SCF85] Schaffer, J.D., *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*, In Proceedings of the 1st International Conference on Genetic Algorithms, Lawrence Erlbaum, 1985, pp. 93-100.

[SCG05] Schamberger, S., Wierum, J., *Partitioning Finite Element Meshes using Space-filling Curves*, Future Generation Computer Systems 21(5), 2005, pp. 759-766.

[SCH00a] Schloegel, K., Karypis, G., Kumar, V., *Parallel Multilevel Algorithms for Multi-constraint Graph Partitioning*, Springer-Verlag Lecture Notes in Computer Science 1900, 2000, pp. 296-310.

[SCH01] Schloegel, K., Karypis, G., Kumar, V., *Wavefront Diffusion and LMSR: Algorithms for Dynamic Repartitioning of Adaptative Meshes*, IEEE Transactions on Parallel and Distributed Systems 12(5), 2001, pp. 451-466.

[SCH02] Schloegel, K., Karypis, G., Kumar, V., *Parallel Static and Dynamic Multi-constraint Graph Partitioning*, Concurrency and Computation: Practice and Experience 14(3), 2002, pp. 219-240.

[SEL06] Selvakkumaran, N., Karypis, G., *Multiobjective Hypergraph Partitioning Algorithms for Cut and Maximum Subdomain Degree Minimization*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(3), 2006, pp. 504-517.

[SER93] Serafini, P., *Simulated Annealing for Multi-Objective Optimization Problems*, In Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application, G.H. Tzeng y col. (eds.), Springer-Verlag, 1993.

[SHE96] Shekhar S., DLiu D.R., *Partitioning Similarity Graphs: A Framework for Declustering Problems*, Information Systems Journal 21(6), 1996, pp. 475-496.

[SHR98] Sherali, H.D., Totlani, R., Loganathan, G.V., *Enhanced Lower Bounds for the Global Optimization of Water Distribution Networks*, Water Resources Research 34(7), 1998, pp. 1831-1841.

[SHR01] Sherali, H.D., Subramanian S., Loganathan, G.V., *Effective Relaxations and Partitioning Schemes for Solving Water Distribution Network Design Problems to Global Optimality*, Journal of Global Optimization 19, 2001, pp. 1-26.

[SHI99] Shimizu, Y., *Multi-Objective Optimization for Site Location Problems through Hybrid Genetic Algorithm with Neural Networks*, Journal of Chemical Engineering of Japan 32(1), 1999, pp. 51–58.

[SIM91] Simon, H.D., *Partitioning of Unstructured Problems for Parallel Processing*, Computing Systems in Engineering 2(2-3), 1991, pp. 135-148.

[SMI99] Smith K., *Neural Networks for Combinatorial Optimization:A Review of More Than a Decade of Research*, INFORMS Journal on Computing 11, 1999, pp. 15-34.

[SNI96] Snir M., Otto S.W., Huss-Lederman S., Walker, D.W., Dongarra J., *MPI The Complete Reference*, MIT Press, 1996.

[SOP04] Soper, A.J., Walshaw, C., Cross, M., *A Combined Evolutionary Search and Multilevel Optimisation Approach to Graph Partitioning*, Journal of Global Optimization 29(2), 2004, pp. 225-241.

[SRI94] Srinivas, N., Deb K., *Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms*, Evolutionary Computation 2(3), 1994, pp. 221–248.

[SRV05] Srivastava, A., Sylvester, D., Blaauw, D., (eds.) *Statistical Analysis and Optimization for VLSI: Timing and Power*, In Series on Integrated Circuits and Systems, Springer, 2005.

[TAL02] Talbi, E-G., *A Taxonomy of Hybrid Metaheuristics*, Journal of Heuristics 8(5), 2002. pp. 541-564.

[TAL06] Talbi, E-G., (ed.) *Parallel Combinatorial Optimization*, In Wiley Series on Parallel and Distributed Computing, John Wiley & Sons, USA, 2006.

[TOD00] Todini, E., *Looped Water Distribution Networks Design using a Resilience Index based Heuristic Approach*, Urban Water 2, 2000, pp. 115-122.

[TOL96] Tollis, I.G., *Optimal Partitioning of Cellular Networks*, In Proceedings of the Conference Record, Converging Technologies for Tomorrow's Applications (ICC'96), vol. 3, 1996, pp. 1377-1381.

[TOR04] Toro-Negro, F., Ortega, J., Ros, E., Mota, S., Paechter, B., Martin, J.M., *PSFGA: Parallel Processing and Evolutionary Computation for Multiobjective Optimisation*, Parallel Computing 30(5-6), 2004, pp. 721-739.

[TSC03] Toscano G., Coello, C.A., *The Micro Genetic Algorithm 2: Towards Online Adaptation in Evolutionary Multiobjective Optimization* Springer-Verlag Lecture Notes in Computer Science 2632, 2003, pp. 252-266

[ULU99] Ulungu, E.L., Teghem, J., Fortemps, P., Tuyytens, D., *MOSA Method: A Tool for Solving Multiobjective Combinatorial Optimization Problems*, Journal of Multi-Criteria Decision Analysis 8(4), 1999, pp. 221-236.

[USEPA] *United States Environmental Protection Agency*, http://www.epa.gov/

[VAI00] Vairavamoorthy, K., Ali, M., *Optimal Design of Water Distribution Systems using Genetic Algorithms*, Computer-Aided Civil and Infraestructure Engineering 15(5), 2000, pp. 374-382.

[VAR97] Varma, K., Narasimhan, S., Bhallamudi, S.M., *Optimal Design of Water Distribution Systems using NLP Method*, Journal of Environmental Engineering (ASCE) 123(4), 1997, pp. 381-388.

[VEL03] Veldhuizen, D.A., Zydallys, J.B., Lamont, G.B., *Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms* IEEE Transactions on Evolutionary Computation 7(2), 2003, pp. 144-173.

[VOB99] Vob, S., Martello, S., Osman, I.H., Roucairol, C., *Meta-Heuristic Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Accademic Publishers, Dordrecht, The Netherlands, 1999.

[VOU98] Voudouris C., Tsang, E.P.K., *Guided Local Search*, European Journal of Operational Research 113(2), 1998, pp. 80-110.

[WAL00] Walshaw, C., Cross, M., *Parallel Optimisation Algorithms for Multilevel Mesh Partitioning*, Parallel Computing 26(12), 2000, pp. 1635-1660.

[WAL01] Walshaw, C., Cross, M., *Multilevel Mesh Partitioning for Heterogeneous Communication Networks*, Future Generation Computer Systems 17(5), 2001, pp. 601-623.

[WAL02] Walshaw, C., Cross, M., *Dynamic Mesh Partitioning and Load Balancing for Parallel Computational Mechanics Codes*, In Computational Mechanics using High Performance Computing, B.H.V. Topping (ed.), Saxe-Coburg Publications, Stirling, UK, 2002, pp. 79-94.

[WAL04] Walshaw, C., *Multilevel Refinement for Combinatorial Optimisation Problems*, Annals of Operations Research 131, 2004, pp. 325-372.

[WAS89] Wasserman, P.D., *Neural Computing: Theory and Practice*, Van Nostrand Reinholdt, New York, 1989.

[ZHA01] Zha, H., He, X., Ding, C., Gu, M., Simon, H., *Bipartite graph partitioning and data clustering*, In Proceedings of the 10th ACM International Conference on Information and Knowledge Management, 2001, pp. 25-31.

[ZIT99] Zitzler, E., Thiele L., *Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach*, IEEE Transactions on Evolutionary Computation 3(4), 1999, pp. 257-271.

[ZIT00] Zitzler, E., Deb, K., Thiele L., *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results*, Evolutionary Computation 8, 2000, pp. 173–195.

[ZIT01] Zitzler, E., Laumanns, M., Thiele L., *SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization*, In Proceedings of the Conference on Evolutionary Methods for Design, Optimisation, and Control, Barcelona, Spain, 2001, pp. 95-100.